工业自动控制系列丛书

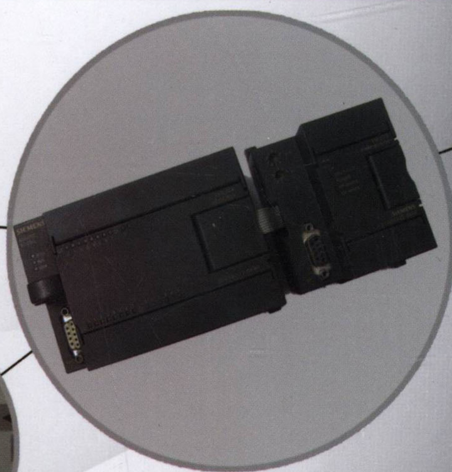
PLC系统编程调试入门

——S7-200问与答

主 编 张运刚 / 副主编 郭武强












机械工业出版社
CHINA MACHINE PRESS

VISIT…

LANZAROTE Caliente.com



张运刚于1989年毕业于华南理工大学物理系，毕业后专职从事PLC系统的设计、选型、编程、安装、调试和维护工作。

2000年开始从事PLC的技术培训工作，2004年开始着手“可编程序系统设计师”技能鉴定的文件和题库工作，是“可编程序系统设计师”业内的开发和领头人。

长期以来，参与过多家企业的技术改造工作，兼多家设备制造企业自动化技术顾问。兼职多家自动化教学设备制造商的技术顾问，长期参与职业技能鉴定、考核设备的研发工作。从2003年以来，受中国自动化学会等多家机构邀请，到全国各省市进行自动化技术专题培训。

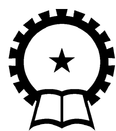
在长期的技术培训实践中，独创出一套PLC编程的方法——“发施号令法”。其特点是入门容易，操作系统编程方便、修改和添加控制功能灵活，减少了编程和系统的工作量。在张运刚讲授的PLC培训课程里，发施号令法会毫无保留地传授给每个学员，让学员在短时间内掌握PLC系统的选型、编程、调试和维护技术。

我们的教学目标是：务实做教育，严谨搞技术。

张运刚的PLC技术支持网址：
www.hnlgplc.com
QQ：200828029

PLC 系统编程调试入门
——S7-200 问与答

主　编　张运刚
副主编　郭武强



机 械 工 业 出 版 社

本书是以西门子 S7-200 PLC 为例，全书内容共有 18 章，简明扼要地介绍了 S7-200 新版编程软件的安装和使用，重点详细地讲述了 S7-200 的软元件属性、基本指令、功能指令、硬件模块和选型。

“我对 PLC 很感兴趣，书看了很多，看后就不明白其意思，更不用说做项目了”这是很多人在咨询 PLC 技术时提到的问题。本书通过问与答的形式，通俗的语言和大量的图解，详细地讲述 S7-200 的应用内容。

本书与众不同的特点是，书中每一个知识点都是可以运行的，通过操作可以看到效果，这就是项目教学法里的小项目。体会操作，从操作中受到启发，从操作中领悟技术要点。

随书配送光盘，光盘内容主要有 SIEMENS. STEP. 7. MicroWIN. V4. 0. SP9. iso 编程软件和 S7-200 的系统手册。

本书可作为工业自动化领域技术人员和爱好者的入门读物，也可作为大中专院校自动化、机电一体化专业学生参考用书，还可以作为 PLC 职业技能培训教材。

图书在版编目（CIP）数据

PLC 系统编程调试入门：S7-200 问与答/张运刚主编.
—北京：机械工业出版社，2013.12
ISBN 978-7-111-45169-3

Ⅰ.①P… Ⅱ.①张… Ⅲ.①plc 技术-程序设计-问题解答
Ⅳ.①TM571.6-44

中国版本图书馆 CIP 数据核字（2013）第 303038 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
策划编辑：林春泉　责任编辑：朱　林
责任印制：张　楠　责任校对：胡艳萍
北京京丰印刷厂印刷
2014 年 1 月第 1 版·第 1 次印刷
184mm×260mm·21 印张·515 千字
0 001—3 000 册
标准书号：ISBN 978-7-111-45169-3
　　　　　ISBN 978-7-89405-247-8（光盘）
定价：58.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
社服务中心：(010)88361066	教材网：http：//www.cmpedu.com
销售一部：(010)68326294	机工官网：http：//www.cmpbook.com
销售二部：(010)88379649	机工官博：http：//weibo.com/cmp1952
读者购书热线：(010)88379203	**封面无防伪标均为盗版**

前　　言

从事工业控制系统相关的工作，不管在开发、选型、编程、调试以及维护，都需要系统的知识和扎实的基本功。特别是在我国，“半桶水”的技术人员太多了，由于竞争非常激烈，作为工程技术人员，需要有高超、精通的技术，这样才能活得轻松，才能活出人生价值来。

“我对 PLC 技术很感兴趣，看了很多书也很用心看但就是看不懂，不知道如何入门，这是我最苦恼的事情”这是很多人在咨询时提到的问题。其实工控的学习，书需要看，但是动手更加重要。本书就是指导、帮助新手如何一边看书一边动手实操，从动手实操中领会、理解和从中学会技能。

本书中的程序实例是作者 10 多年学术研究、实际工程实践和亲自教学的结晶。本书最大的亮点就是作者将独特的编程逻辑和调试方法介绍给读者。其次一个特点就是少说无为的理论，结合工程实际，多说解决问题的方法，这些实例可以当“饭”吃，少做改动就可以套用于现场的控制任务。

本书内容来源于实际工程，并经过作者在长期 PLC 的培训教学中，将这些内容有机地转移到生活空间的边缘，让较少接触工控实际的读者也备感亲切，容易读懂。书中的语言非常口语化，避免生硬的专业语句，使新手很容易阅读和读懂。

整套丛书按照品牌不同分有很多本，但是内容风格一致，使读者能够轻松阅读，读完整套丛书后基本可以轻松达到举一反三、一通百通、灵活应用的效果。

由于编者水平有限，书中错漏有所难免，恳请广大读者批评指正，衷心感谢！

作者

2013 年 11 月

目　　录

前言
第 1 章　软件安装与使用 …… 1
1.1　软件如何安装 …… 1
问 1：安装西门子 S7-200 的编程软件时，计算机需要具备什么系统？ …… 1
问 2：西门子 S7-200 的编程软件如何安装？ …… 1
1.2　程序的编写 …… 7
问 1：在计算机里如何打开 V4.0 STEP 7 MicroWIN SP9 编程软件？ …… 8
问 2：打开编程界面的语言是英文，能否转为中文界面？ …… 9
问 3：在编程界面怎样输入指令和软元件？ …… 9
问 4：在编程时，怎样画垂直线和水平线？ …… 13
问 5：在编程时，怎样插入列和行？ …… 15
问 6：在编程时，怎样插入和删除网络？ …… 20
问 7：在编程时，怎样添加和删除程序？ …… 20
1.3　通信和监控 …… 23
问 1：V4.0 STEP 7 MicroWIN SP9 编程软件怎样才能与 CPU 通信？ …… 23
问 2：怎样下载程序到 CPU？ …… 25
问 3：如何进入程序监控状态？ …… 26
问 4：CPU 面板上有一个开关，旁边有 RUN/TERM/STOP 字样，分别是什么意思？ …… 26
1.4　程序的修改和错误处理 …… 29
问 1：在编写程序时，如果发现指令错了，怎样更改指令？ …… 29
问 2：在编写程序时，如果发现软元件错了，怎样更改软元件？ …… 30
问 3：在编写程序过程中，出现编译错误如何处理？ …… 33
问 4：下载程序时，出现错误如何处理？ …… 33
1.5　程序的注释和项目保存 …… 35
问 1：编程时使用中文注释可以吗？如何使用？ …… 36
问 2：项目如何保存在计算机硬盘里？ …… 41
第 2 章　S7-200 的软元件 …… 44
2.1　I/Q 输入/输出 …… 44
问：在程序中看到有 I 和 Q 的符号，如何理解 I 和 Q？ …… 44
2.2　M/S 中间继电器/状态继电器 …… 46
问：在程序中看到有 M、SM 和 S 的符号，如何理解这些软元件？ …… 46
2.3　V/L 数据存储器/临时寄存器 …… 48
问：S7-200 程序中的 V 和 L 符号代表什么符号？如何理解它们的属性？ …… 48
2.4　常量和数制 …… 50
问：S7-200 支持什么进制数据？支持中文字符类型吗？怎样理解这些数据和符号？ …… 50
第 3 章　基本指令 …… 53
3.1　一个开关驱动一个输出 …… 53

问 1：一个开关控制一个输出的程序怎样编写？ …… 53
问 2：如何调试和监控程序？ …… 53
3.2　一个开关置位/复位输出 …… 55
问 1：一个开关控制一个输出，使用“SET”和“RST”指令的程序怎样编写？ …… 55
问 2：如何调试和监控程序？ …… 55
3.3　一个开关脉冲沿置位/复位输出 …… 57
问：一个开关使用脉冲沿控制一个输出，程序怎样编写？如何调试和监控？ …… 57
3.4　启动按钮/停止按钮/保持/驱动输出 …… 59
问 1：使用两个按钮控制一个输出的程序怎样编写？ …… 59
问 2：如何调试和监控程序？ …… 59
3.5　启动按钮/停止按钮/置位/复位输出 …… 62
问 1：两个按钮控制一个输出，使用“SET”和“RST”指令 RST 优先的程序怎样编写？ …… 62
问 2：如何调试和监控程序？ …… 62
问 3：两个按钮控制一个输出，使用“SET”和“RST”指令 SET 优先的程序怎样编写？ …… 64
问 4：如何调试和监控程序？ …… 64
3.6　启动按钮/停止按钮脉冲沿/置位/复位输出 …… 66
问 1：两个按钮的脉冲沿控制一个输出，使用“SET”和“RST”指令 RST 优先的程序怎样编写？ …… 66
问 2：如何调试和监控程序？ …… 66
问 3：两个按钮的脉冲沿控制一个输出，使用“SET”和“RST”指令 SET 优先的程序怎样编写？ …… 68
问 4：如何调试和监控程序？ …… 69
3.7　一个按钮控制一个输出 …… 70
问：控制逻辑如图 3-28 所示，一个按钮控制一个输出的程序怎样编写？ …… 70
3.8　步进阶梯指令 …… 72
问 1：什么时候使用步进阶梯指令编程有优势？ …… 72
问 2：如何理解步的状态？ …… 72
问 3：步进阶梯指令怎样编程？ …… 73
第 4 章　定时器和系统时钟 …… 80
4.1　TON …… 80
问 1：TON 型定时器的定时规律怎样？ …… 80
问 2：TON 型定时器有哪些？ …… 80
问 3：如何使用 TON 型定时器？ …… 80
4.2　TOF …… 82
问 1：TOF 型定时器的定时规律怎样？ …… 82
问 2：TOF 型定时器有哪些？ …… 82
问 3：如何使用 TOF 型定时器？ …… 82
4.3　TONR …… 84
问 1：TONR 型定时器的定时规律怎样？ …… 84
问 2：TONR 型定时器有哪些？ …… 84
问 3：如何使用 TONR 型定时器？ …… 84

4.4　BGN _ ITIME/CAL _ ITIME …… 86
问 1：BGN_ ITIME 和 CAL_ ITIME 指令基本动作是什么？ …… 86
问 2：BGN_ITIME 和 CAL_ITIME 指令如何使用？ …… 86
4.5　READ _ RTC/READ _ RTCX/SET _ RTC/SET _ RTCX …… 86
问：S7-200 的系统时钟怎样校对时间？如何读取系统时钟？ …… 87
第 5 章　计数器 …… 89
5.1　CTU …… 89
问：CTU 计数规律是什么？怎样探讨 CTU 计数器规律？ …… 89
5.2　CTD …… 91
问：CTD 计数规律是什么？怎样探讨 CTD 计数器规律？ …… 91
5.3　CTUD …… 93
问：CTUD 计数规律是什么？怎样探讨 CTUD 计数器规律？ …… 93
第 6 章　传送指令 …… 95
6.1　MOV _ B/W/DW/R …… 95
问 1：MOV 指令基本功能是什么？ …… 95
问 2：MOV 指令样式是怎样的？ …… 95
问 3：如何应用 MOV 指令？ …… 96
问 4：在应用 MOV 指令时需要注意些什么？ …… 99
6.2　BLKMOV _ B/W/DW …… 99
问 1：BLKMOV 指令基本功能是什么？ …… 99
问 2：BLKMOV 指令样式是怎样的？ …… 100
问 3：如何应用 BLKMOV 指令？ …… 100
问 4：在应用 BLKMOV 指令时需要注意些什么？ …… 101
6.3　FILL _ N …… 101
问 1：FILL _ N 指令基本功能是什么？ …… 102
问 2：FILL _ N 指令样式是怎样的？ …… 102
问 3：如何应用 FILL _ N 指令？ …… 102
问 4：在应用 FILL _ N 指令时需要注意些什么？ …… 103
6.4　SWAP …… 103
问 1：SWAP 指令基本功能是什么？ …… 103
问 2：SWAP 指令样式是怎样的？ …… 103
问 3：如何应用 SWAP 指令？ …… 103
问 4：在应用 SWAP 指令时需要注意些什么？ …… 104
6.5　INV _ B/W/DW …… 104
问 1：INV 指令基本功能是什么？ …… 104
问 2：INV 指令样式是怎样的？ …… 104
问 3：如何应用 INV 指令？ …… 105
问 4：在应用 INV 指令时需要注意些什么？ …… 107
6.6　MOV _ BIR/MOV _ BIW …… 107
问 1：MOV _ BIR 和 MOV _ BIW 指令基本功能是什么？ …… 107
问 2：MOV _ BIR 和 MOV _ BIW 指令样式是怎样的？ …… 107
问 3：如何应用 MOV _ BIR 和 MOV _ BIW 指令？ …… 108
问 4：在应用 MOV _ BIR 和 MOV _ BIW 指令时需要注意些什么？ …… 111

第 7 章　触点比较指令 …… 113
7.1　数值比较 =/≠/>/</≥/≤ …… 113
问 1：数值比较指令基本功能是什么？ …… 113
问 2：数值比较指令样式是怎样的？ …… 113
问 3：如何应用数值比较指令？ …… 114
问 4：在应用数值比较指令时需要注意些什么？ …… 118
7.2　字符串比较 =/≠ …… 120
问 1：字符串比较指令基本功能是什么？ …… 120
问 2：字符串比较指令样式是怎样的？ …… 120
问 3：如何应用字符串比较指令？ …… 120
问 4：在应用字符串比较指令时需要注意些什么？ …… 120
第 8 章　数学运算和转换指令 …… 122
8.1　整数运算 …… 122
问 1：整数的特征是什么？ …… 122
问 2：整数运算指令样式是怎样的？ …… 122
问 3：整数运算指令基本运算规律怎样？ …… 125
问 4：在应用整数运算指令时需要注意些什么？ …… 135
8.2　小数运算 …… 135
问 1：小数的特征是什么？ …… 135
问 2：小数运算指令样式是怎样的？ …… 136
问 3：小数运算指令基本运算规律怎样应用？ …… 138
问 4：在应用小数运算指令时需要注意些什么？ …… 147
8.3　数值类型转换 B⟷I⟷DI⟷R …… 148
问 1：能否计算有 8 位字节、16 位整数、32 位整数和小数的混合运算？ …… 148
问 2：数值转换指令样式是怎样的？ …… 148
问 3：这些转换指令基本规律怎样？ …… 149
问 4：在应用数值转换指令时需要注意些什么？ …… 151
8.4　BCD 码和七段码转换 …… 152
问 1：以前没有触摸屏人机界面时能否实现人机界面功能？ …… 152
问 2：人机界面数据转换指令样式是怎样的？ …… 152
问 3：人机界面数据转换指令基本规律怎样？ …… 152
问 4：在应用人机界面数据转换指令时需要注意些什么？ …… 156
8.5　字符（串）转换 …… 157
问 1：S7-200 CPU 支持字符功能吗？支持字符串功能吗？支持中文字符串吗？ …… 157
问 2：这些字符和字符串指令样式是怎样的？ …… 157
问 3：字符和字符串指令基本规律怎样？ …… 160
问 4：在应用字符和字符串指令时需要注意些什么？ …… 176
8.6　编码/译码转换 …… 176
问 1：什么时候会使用到编码解码指令？ …… 176
问 2：编码解码指令样式是怎样的？ …… 177
问 3：编码解码指令基本规律怎样？ …… 177
问 4：在应用编码解码指令时需要注意些什么？ …… 181
第 9 章　加一减一逻辑指令 …… 182

9.1 INC _ B/W/DW …… 182
问 1：INC 指令基本功能是什么？ …… 182
问 2：INC 指令样式是怎样的？ …… 182
问 3：如何应用 INC 指令？ …… 182
问 4：在应用 INC 指令时需要注意些什么？ …… 186
9.2 DEC _ B/W/DW …… 187
问 1：DEC 指令基本功能是什么？ …… 187
问 2：DEC 指令样式是怎样的？ …… 187
问 3：如何应用 DEC 指令？ …… 188
问 4：在应用 DEC 指令时需要注意些什么？ …… 193
9.3 几种加减法有何不同 …… 193
问 1：编程时常用的加/减法逻辑有几种？ …… 193
问 2：这些加/减法各有什么特点？ …… 193
第 10 章 循环移位表逻辑指令 …… 197
10.1 SHL/SHR _ B/W/DW …… 197
问 1：移位指令基本功能是什么？ …… 197
问 2：移位指令样式是怎样的？ …… 197
问 3：如何应用移位指令？ …… 198
问 4：在应用移位指令时需要注意些什么？ …… 201
10.2 ROL/ROR _ B/W/DW …… 201
问 1：循环指令基本功能是什么？ …… 201
问 2：循环指令样式是怎样的？ …… 201
问 3：如何应用循环指令？ …… 202
问 4：在应用循环指令时需要注意些什么？ …… 205
10.3 SHRB …… 205
问 1：SHRB 指令基本功能是什么？ …… 205
问 2：SHRB 指令样式是怎样的？ …… 205
问 3：如何应用 SHRB 指令？ …… 206
问 4：在应用 SHRB 指令时需要注意些什么？ …… 207
10.4 表指令 …… 207
问 1：表指令基本功能是什么？ …… 208
问 2：表指令样式是怎样的？ …… 208
问 3：如何应用表指令？ …… 208
问 4：在应用表指令时需要注意些什么？ …… 215
第 11 章 与或异或逻辑指令 …… 216
11.1 WAND _ B/W/DW …… 216
问 1：WADN 与逻辑指令基本功能是什么？ …… 216
问 2：WADN 与逻辑指令样式是怎样的？ …… 216
问 3：如何应用 WADN 与逻辑指令？ …… 217
问 4：在应用 WADN 与逻辑指令时需要注意些什么？ …… 219
11.2 WOR _ B/W/DW …… 220
问 1：WOR 或逻辑指令基本功能是什么？ …… 220
问 2：WOR 或逻辑指令样式是怎样的？ …… 220

问 3：如何应用或逻辑指令？ …… 220
问 4：在应用 WOR 或逻辑指令时需要注意些什么？ …… 224
11.3　WXOR _ B/W/DW …… 224
问 1：WXOR 异或逻辑指令基本功能是什么？ …… 224
问 2：WXOR 异或逻辑指令样式是怎样的？ …… 224
问 3：如何应用 WXOR 异或逻辑指令？ …… 224
问 4：在应用 WXOR 异或逻辑指令时需要注意些什么？ …… 227
第 12 章　程序控制指令 …… 229
12.1　JMP/LBL …… 229
问 1：跳转指令基本动作是什么？ …… 229
问 2：跳转指令样式是怎样的？ …… 229
问 3：如何应用跳转指令？ …… 229
问 4：在应用跳转指令时需要注意些什么？ …… 234
12.2　ROR/NEXT …… 236
问 1：FOR _ NEXT 指令基本动作是什么？ …… 236
问 2：FOR _ NEXT 指令样式是怎样的？ …… 236
问 3：如何应用 FOR _ NEXT 指令？ …… 236
问 4：在应用 FOR _ NEXT 指令时需要注意些什么？ …… 237
12.3　END/STOP/WDR …… 238
问 1：END 指令基本动作是什么？ …… 239
问 2：如何应用 END 指令？ …… 239
问 3：在应用 END 指令时需要注意些什么？ …… 239
问 4：STOP 指令基本动作是什么？ …… 240
问 5：如何应用 STOP 指令？ …… 240
问 6：在应用 STOP 指令时需要注意些什么？ …… 240
问 7：WDR 指令基本动作是什么？ …… 240
问 8：如何应用 WDR 指令？ …… 241
问 9：在应用 WDR 指令时需要注意些什么？ …… 241
第 13 章　子程序中断程序库指令 …… 243
13.1　CALL/RET …… 243
问 1：S7-200 CPU 中有几种程序？ …… 243
问 2：这些程序什么时候运行？ …… 243
问 3：S7-200 CPU 中有多少子程序？ …… 243
问 4：怎样使用子程序？ …… 243
问 5：使用子程序需要注意些什么？ …… 248
13.2　中断程序 …… 249
问 1：S7-200 CPU 中的中断程序有几种？ …… 249
问 2：中断指令有哪些？ …… 251
问 3：怎样使用中断程序？ …… 251
问 4：使用中断程序时需要注意些什么？ …… 254
第 14 章　高速计数器 …… 256
问 1：高速计数器指令有哪些？ …… 256

问 2：S7-200 支持几个高速计数器？ …… 256
问 3：S7-200 高速计数器各种计数模式的输入 I 分配情况如何？ …… 257
问 4：这些高速计数器控制字分配情况怎样？ …… 257
问 5：这些高速计数器定义初值和目标值情况怎样？ …… 258
问 6：监控这些高速计数器的状态字分配情况怎样？ …… 258
问 7：如何理解高速计数器和应用高速计数器？ …… 259
第 15 章　脉冲输出指令 …… 267
问 1：S7-200 CPU 可以发几路脉冲？ …… 267
问 2：PWM 与 PTO 脉冲有什么特征？ …… 267
问 3：脉冲输出控制字和状态字是什么？ …… 268
问 4：如何应用发 PWM 脉冲？ …… 269
问 5：如何应用发单段 PTO 脉冲？ …… 270
问 6：如何应用发多段 PTO 脉冲？ …… 273
第 16 章　累加器和指针 …… 280
问 1：S7-200 CPU 有几个累加器？ …… 280
问 2：累加器如何使用？ …… 280
问 3：S7-200 CPU 在程序中有几种寻址方式？ …… 280
问 4：间接寻址俗称为指针寻址，如何使用？ …… 280
第 17 章　扩展模块与模拟量 …… 285
17. 1　模块和地址 …… 285
问 1：S7-200 的 CPU 有哪些？ …… 285
问 2：S7-200 的 CPU 技术规范怎样？ …… 286
问 3：常用的数字量扩展模块有哪些？ …… 287
问 4：常用的模拟量扩展模块有哪些？ …… 288
问 5：S7-200 的特殊功能模块有哪些？ …… 289
问 6：扩展模块的地址分配规律是什么？ …… 291
17. 2　模拟量表示法 …… 292
问 1：工业标准的模拟量是什么？ …… 293
问 2：PLC 对模拟量处理流程是怎样的？ …… 293
问 3：S7-200 的 CPU 怎样表达模拟值？ …… 293
17. 3　模拟量控制算法 …… 296
问 1：模拟量输出控制算法有哪些？ …… 296
问 2：这些算法在实际工程中如何应用？ …… 296
第 18 章　通信指令 …… 305
18. 1　SET _ ADDR/GET _ ADDR …… 305
问 1：S7-200 CPU 的 PORT0 和 PORT1 通信口地址在运行时可以更改吗，如何更改？ …… 305
问 2：如果可以更改，怎样知道已经更改为多少？ …… 306
18. 2　NETW/NETR …… 306
问 1：有几台 S7-200 的 CPU 在一个不大的车间内需要互相通信，
用什么方式最容易又省成本？ …… 306
问 2：如何理解 PPI 通信？ …… 307
问 3：怎样实现 PPI 网络通信？ …… 308

18.3　XMT/RCV …… 312
问1：S7-200 CPU 支持 RS485 自由协议通信吗？ …… 312
问2：如何理解 RS485 自由协议通信？ …… 312
问3：怎样实现 RS485 自由协议通信？ …… 315

第 1 章　软件安装与使用

1.1　软件如何安装

> 问：安装西门子 S7-200 的编程软件时，计算机需要具备什么系统？
> 西门子 S7-200 的编程软件如何安装？

问 1：安装西门子 S7-200 的编程软件时，计算机需要具备什么系统？

答：现在计算机流行的系统有 Windows XP SP3 和 WIN7 等系统，这些非家庭版的系统一般都可以安装，但是为了使用得更顺手些，最好选择纯净安装版系统。其中 Windows XP SP3 和 WIN7 的 32 位系统兼容性会好些，支持绝大多数的工控软件，当然 V4.0 STEP 7 MicroWIN SP9 新版软件安装在 WIN7 的 64 位系统也非常好用。

问 2：西门子 S7-200 的编程软件如何安装？

答：安装 V4.0 STEP 7 MicroWIN SP9 软件在 Windows XP SP3 系统上很简单，找到并双击“Setup.exe”安装文件，马上开始安装，按照向导一步一步地单击“next”或者“确认”就可以安装完毕。

下面详细介绍在 WIN7 系统安装 V4.0 STEP 7 MicroWIN SP9 软件的步骤和方法。

启动电脑进入 WIN7 系统，如图 1-1 所示。



图 1-1　WIN7 界面

首先在 C 盘手动创建目录 C:\Program Files(x86)\Siemens\STEP 7MicroWIN V4.0\bin，然后把光盘里面的 microwin.exe 老版本的执行文件复制到刚建立的 bin 目录下。

把载有 SIEMENS. STEP. 7. MicroWIN. V4. 0. SP9. iso 编程软件的光盘放进电脑的光驱里，在 WIN7 桌面上双击“我的计算机”图标，出现如图 1-2 所示的界面，双击光驱图标打开光驱。



图 1-2　打开光驱

在光盘里找到 SIEMENS. STEP. 7. MicroWIN. V4. 0. SP9. iso 编程软件，双击“SIEMENS. STEP. 7. MicroWIN. V4. 0. SP9. iso”图标，打开安装界面，然后在出现的画面中双击“setup. exe”图标，如图 1-3 所示。

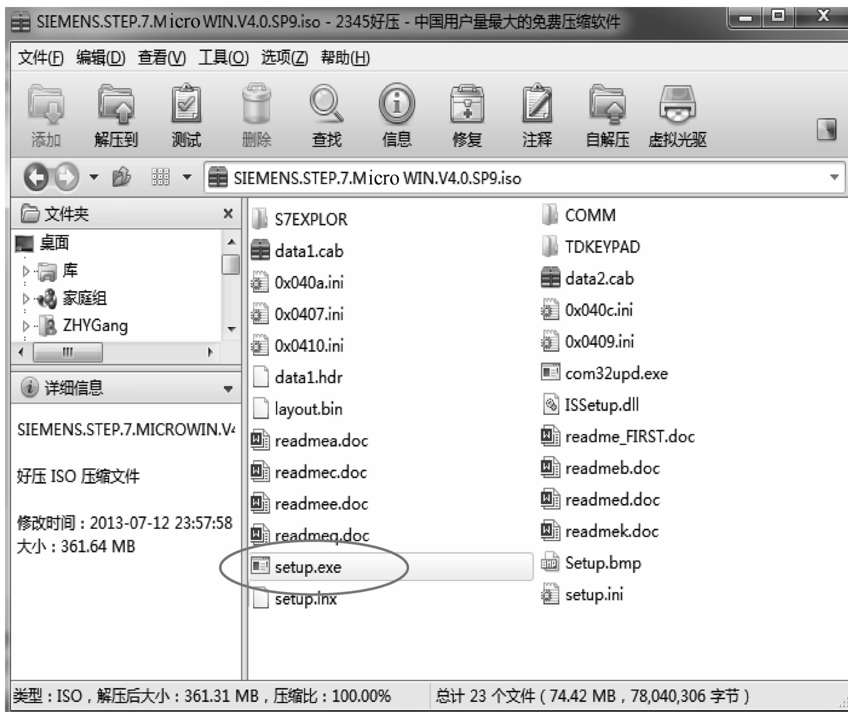


图 1-3　STEP 7-MicroWIN 安装文件

在安装过程中出现“选择设置语言”画面，选择“English”，单击“Next”按钮，如图1-4所示。然后出现安装过程画面，如图1-5所示，这时请耐心等待。

当出现如图1-6所示的画面时，单击“确定”按钮，继续安装。

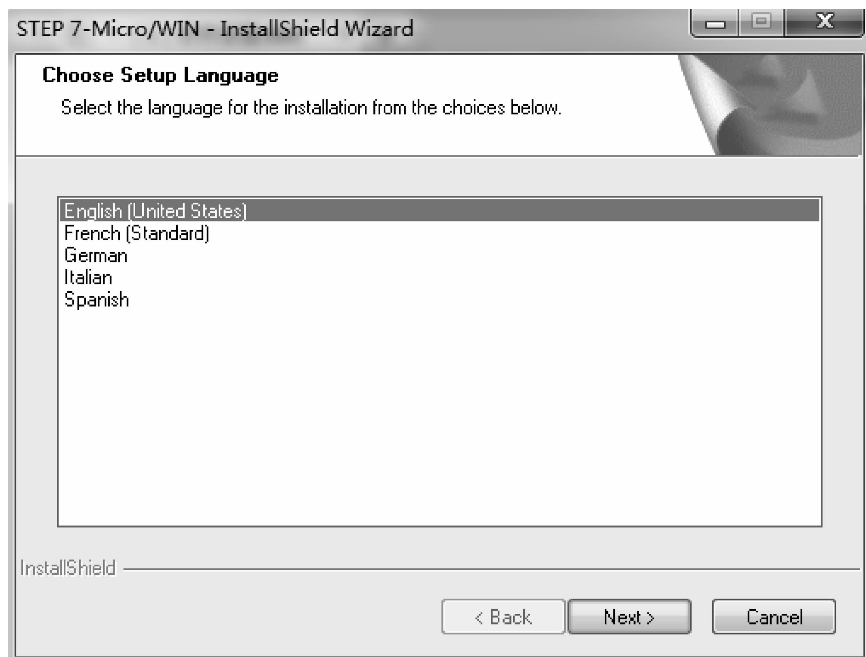


图1-4　选择语言

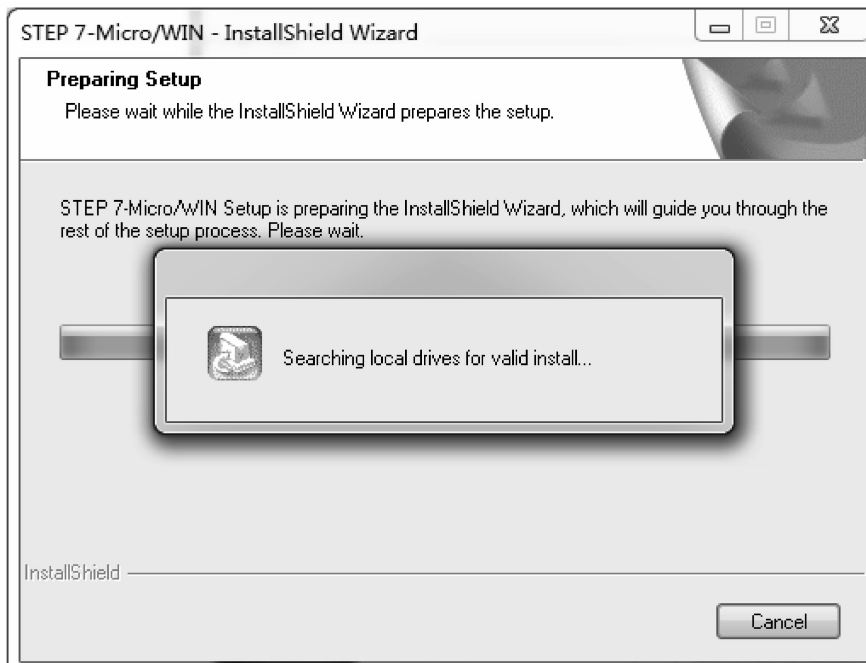


图1-5　安装过程

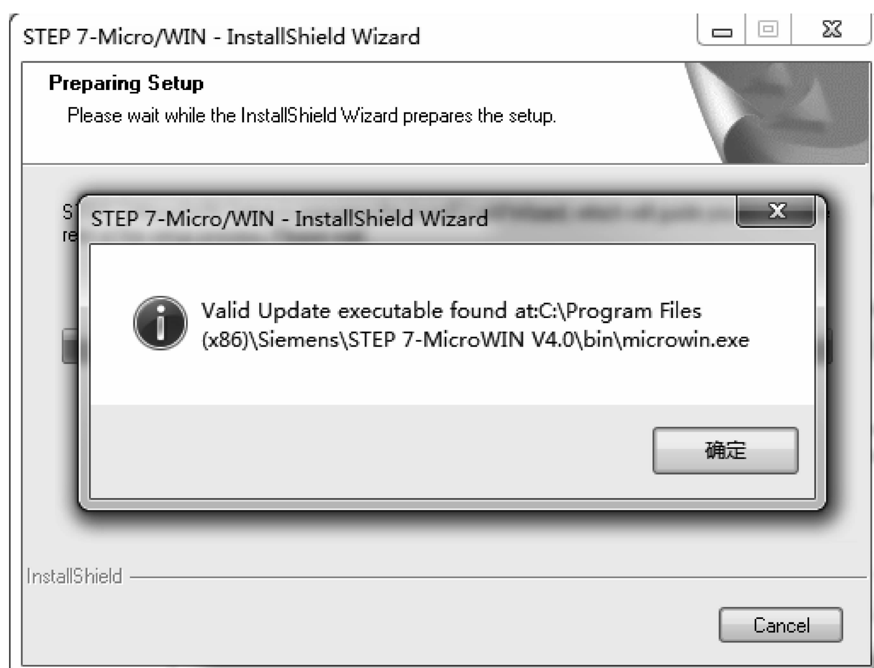


图 1-6　安装向导确定

随后按照顺序出现如图 1-7 ~ 图 1-9 的安装向导画面，单击“Next”或者“Yes”按钮即可。

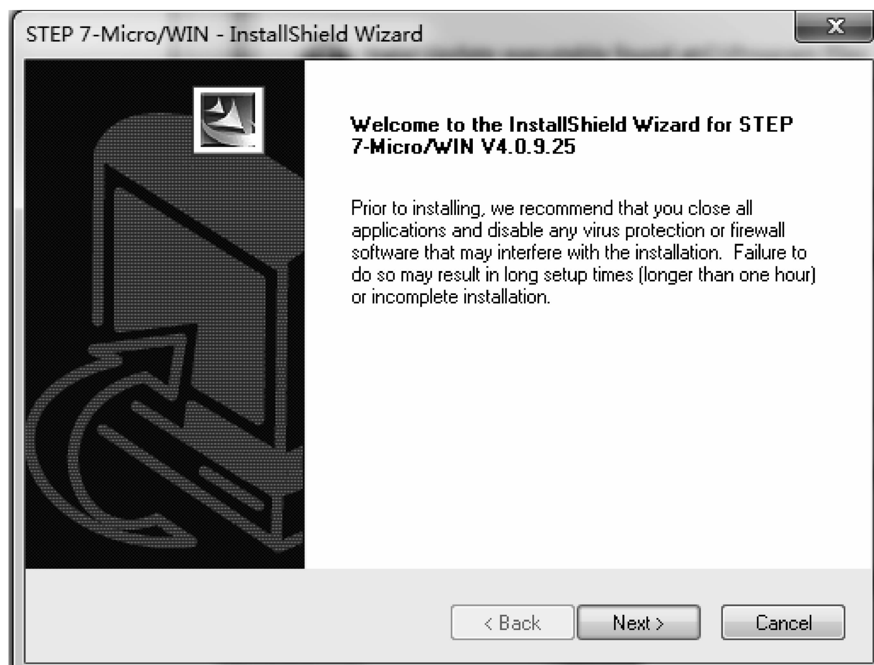


图 1-7　安装向导画面

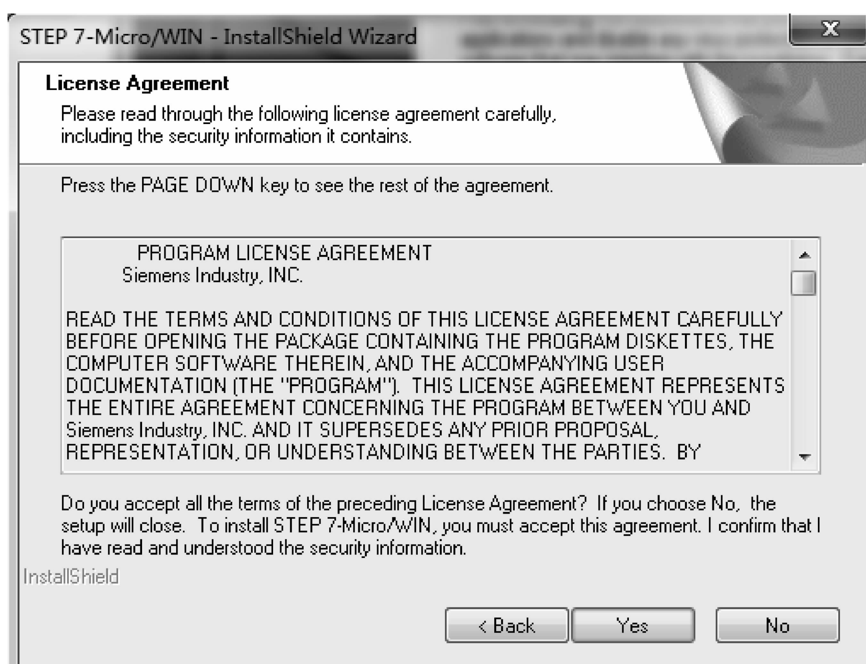


图 1-8　安装向导画面

一般选择安装路径及名称都是默认路径和名称，然后单击“Next”按钮继续安装，如图 1-9 所示。

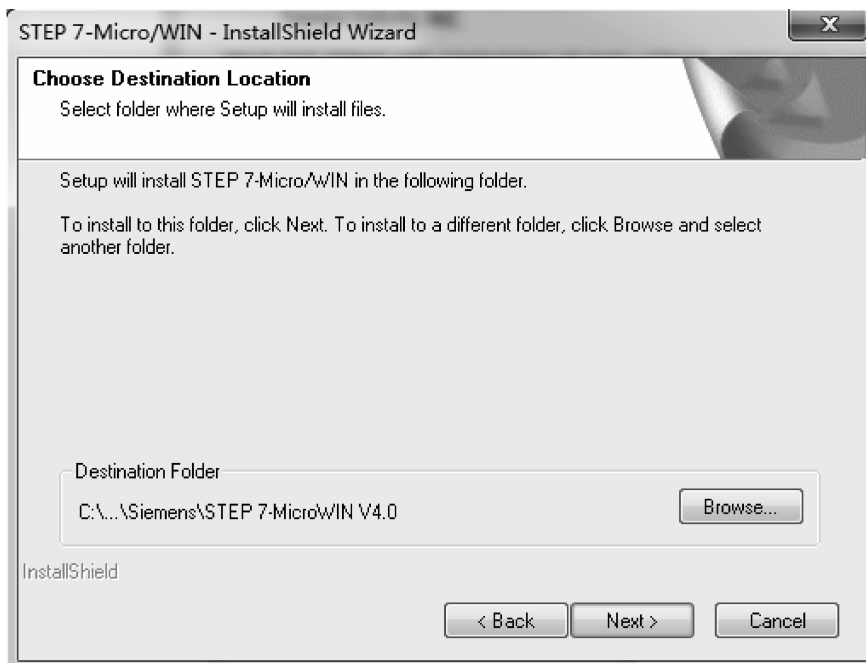


图 1-9　确认路径和名称

当向导出现如图 1-10 所示提示信息时，系统提示卸载旧版，是刚才在 bin 目录放置旧版 microwin. exe 的原因，进入 bin 目录，删除旧版 microwin. exe 的文件，然后单击图 1-10 中的“确定”按钮，继续安装。

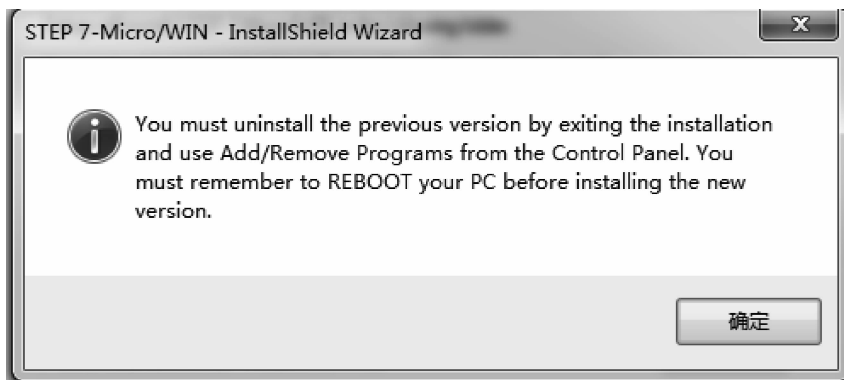


图 1-10　提示删除信息

安装进行中，请等待，如图 1-11 所示。

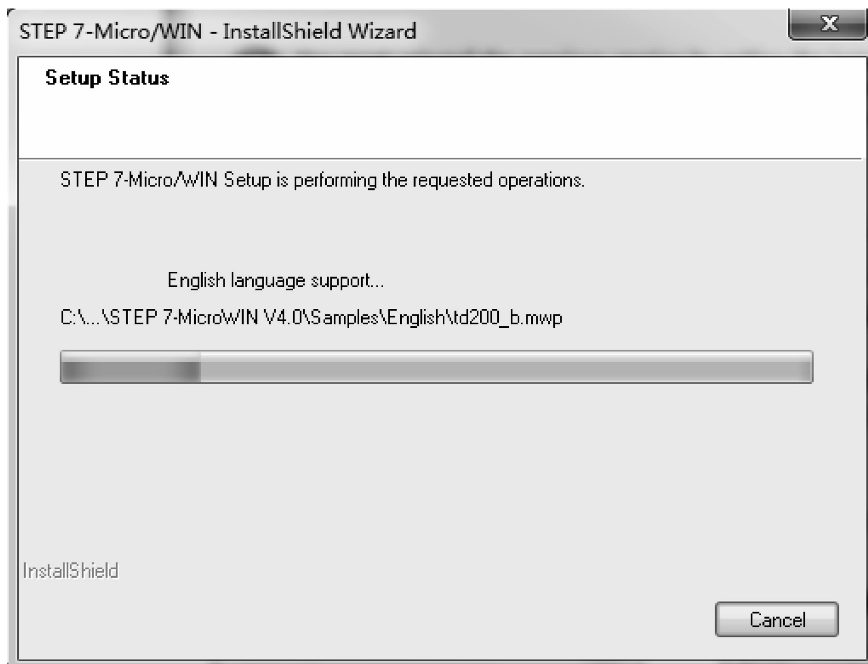


图 1-11　安装过程中

在安装过程中如果弹出如图 1-12 所示信息，意思是软件帮助信息由于缺少系统补丁文件而不支持，需要安装“WinHlp32. exe”文件才能正常使用软件帮助信息。暂时单击图 1-12 中的“确定”按钮，等软件安装完成后再安装“WinHlp32. exe”文件。

安装完成，单击“Finish”按钮，如图 1-13 所示。



图 1-12　帮助提示信息

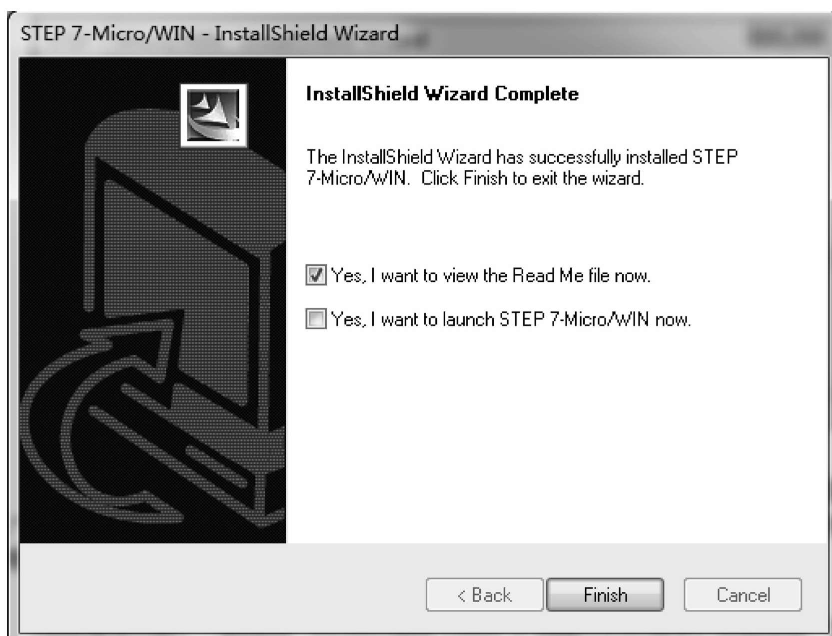


图 1-13　安装完成

1.2　程序的编写

问：在计算机里如何打开 V4.0 STEP 7 MicroWIN SP9 编程软件？
打开编程界面的语言是英文，能否转为中文界面？
在编程界面怎样输入指令和软元件？
在编程时，怎样画垂直线和水平线？
在编程时，怎样插入列和行？
在编程时，怎样插入和删除网络？
在编程时，怎样添加和删除程序？

问 1：在计算机里如何打开 V4.0 STEP 7 MicroWIN SP9 编程软件?

答：一般通过两种方法打开 V4.0 STEP 7 MicroWIN SP9 编程软件：

通过所安装的路径打开 V4.0 STEP 7 MicroWIN SP9 编程软件，本书演示默认安装路径 C:\Program Files (x86)\Siemens\STEP 7-MicroWIN V4.0\bin，双击 bin 文件夹里的“microwin.exe”，打开 V4.0 STEP 7 MicroWIN SP9 编程软件的编程界面，如图 1-14 所示。



图 1-14　从安装路径打开编程界面

在桌面上选中 V4.0 STEP 7 MicroWIN SP9 编程软件的快捷图标，按动鼠标右键，出现下拉菜单，单击下拉菜单中的“打开”或者在桌面上用鼠标双击 V4.0 STEP 7 MicroWIN SP9 编程软件的快捷图标，如图 1-15 所示，即可打开编程软件。



图 1-15　从桌面的快捷图标打开编程界面

问 2：打开编程界面的语言是英文，能否转为中文界面?

答：第一次打开编程界面是英文的，如果需要显示其他语言，可以进行语言切换。切换方法是首先打开编程界面，在文件栏单击“Tools”→“Options”，也就是打开“工具”栏中的“选项”，如图 1-16 所示。在打开的 Options 界面中，选中“General”栏目，在语言选择框里显示可以切换的语言如图 1-17 所示，原则上可以随意切换，比如选择“Chinese”，然后单击“OK”按钮，提示关闭软件，重新打开编程软件的界面就是中文界面了，如图 1-18 所示。

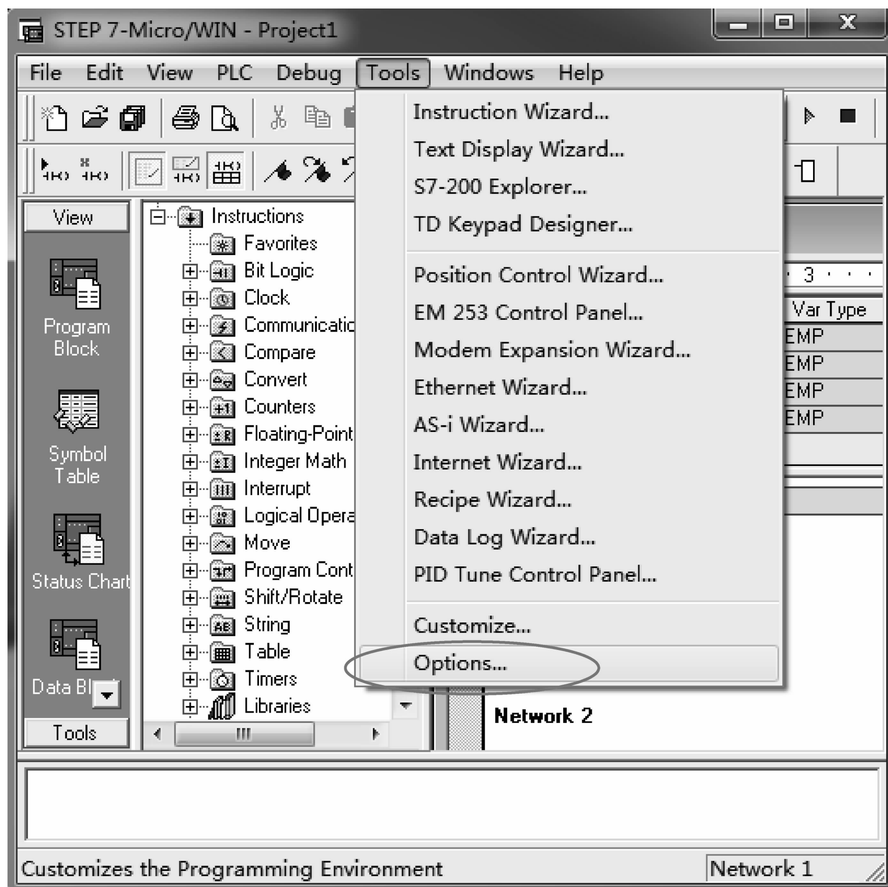


图 1-16　打开工具栏中的选项

问 3：在编程界面怎样输入指令和软元件?

答：在指令树里选中需要输入的指令，如图 1-19 所示的“A”，将指令拖曳至所需的位置如“B”，指令就放置在指定的 B 位置了；也可以先用鼠标在需要放置指令的地方单击一图 1-20 中的“C”点，然后双击指令树中要输入的指令，那么指令将自动出现在需要的 C 位置上。输入的指令如图 1-21 所示。

在图 1-20 中，用鼠标单击指令上的“?? . ?”，可以输入元件的地址，如“I0. 0”和“Q0. 0”，如图 1-22 所示。

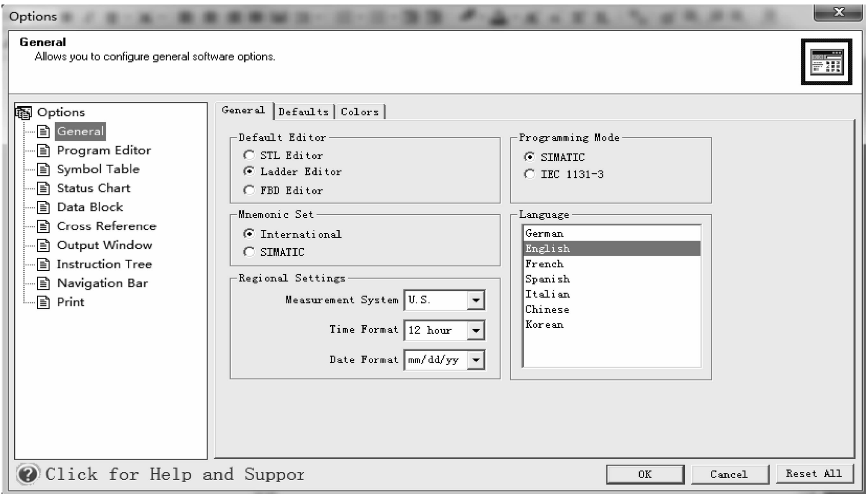


图 1-17 选择语言界面

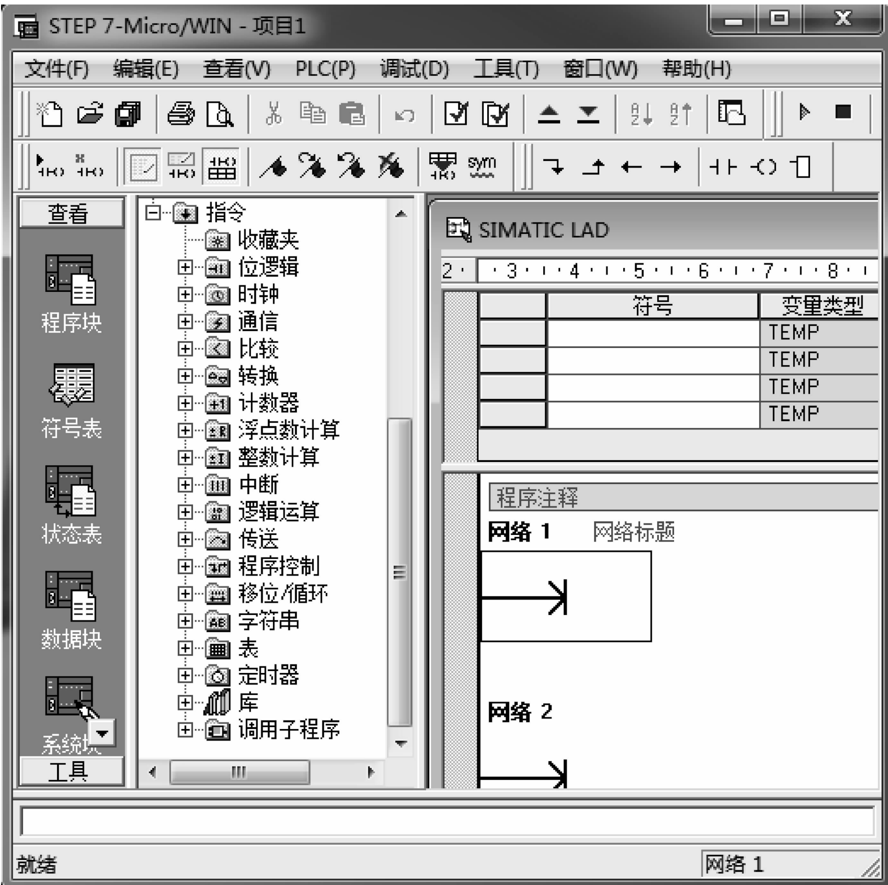


图 1-18 中文编程界面

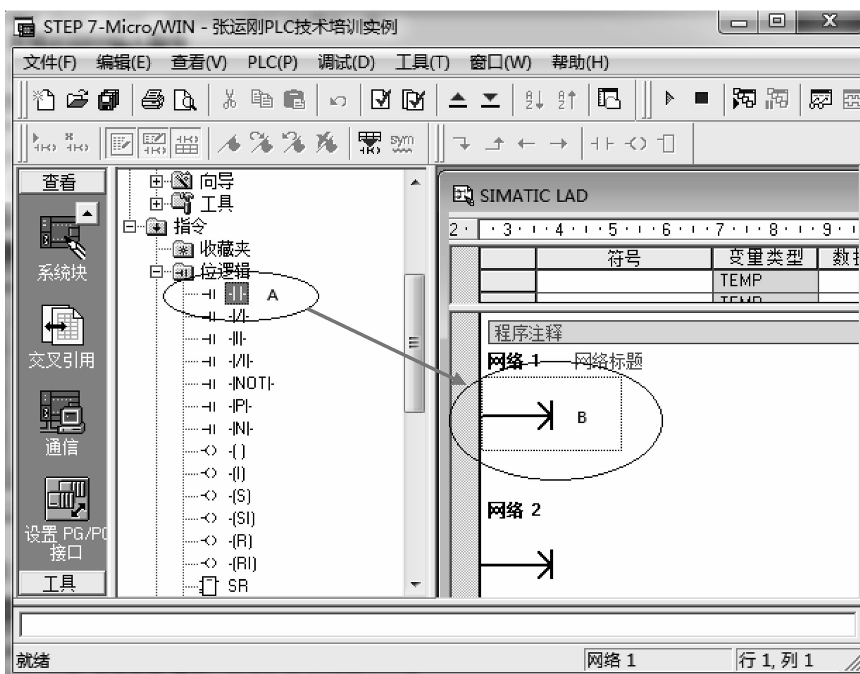


图 1-19　拖曳法输入指令

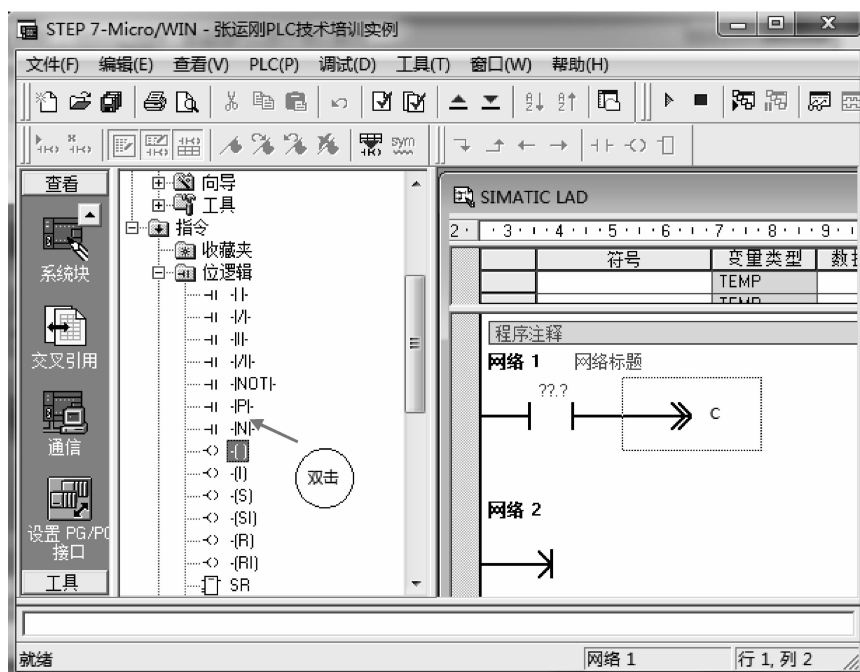


图 1-20　双击法输入指令

图 1-21　输入的指令

图 1-22　输入软元件地址

问 4：在编程时，怎样画垂直线和水平线？

答：画垂直线。如果要求输入如图 1-23 所示的程序，可以先按照输入指令的办法输入如图 1-24 所示的程序，然后把光标放在 I0.1 的常开触点指令上，如图 1-25 所示，用鼠标单击“↑”符号即可画好向上的垂直线。或者把光标放在 I0.0 的常开触点指令上，然后用鼠标单击“↓”符号即可画出向下的垂直线。

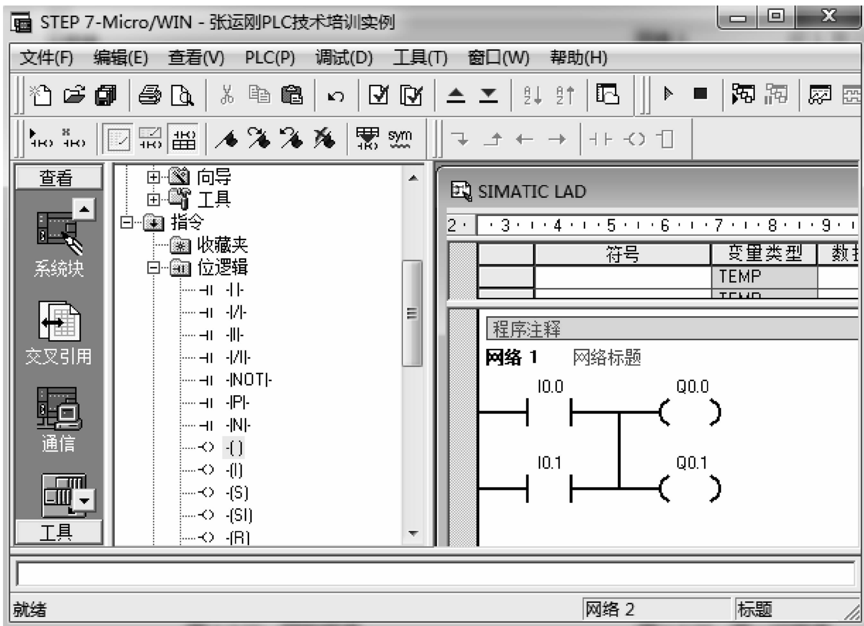


图 1-23　画垂直线目标程序

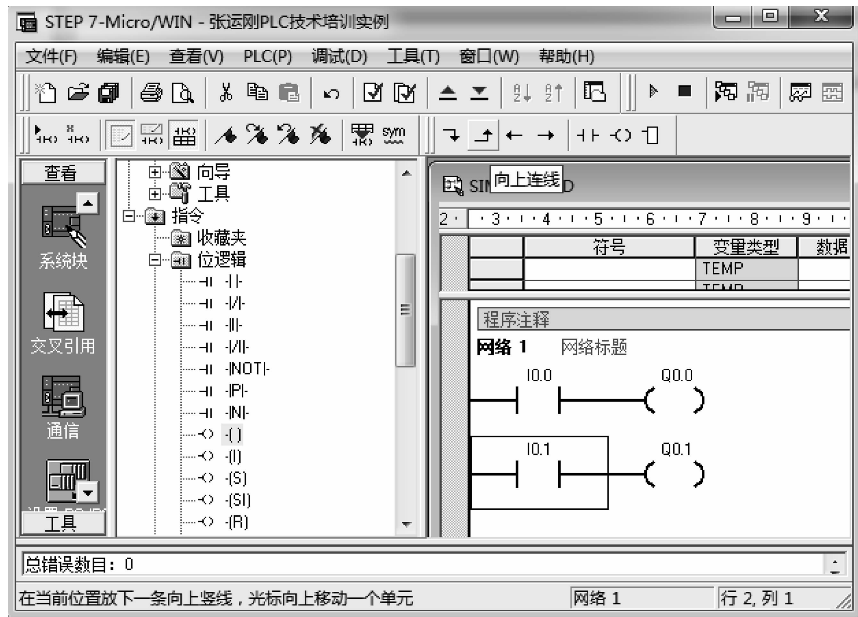


图 1-24　光标位置

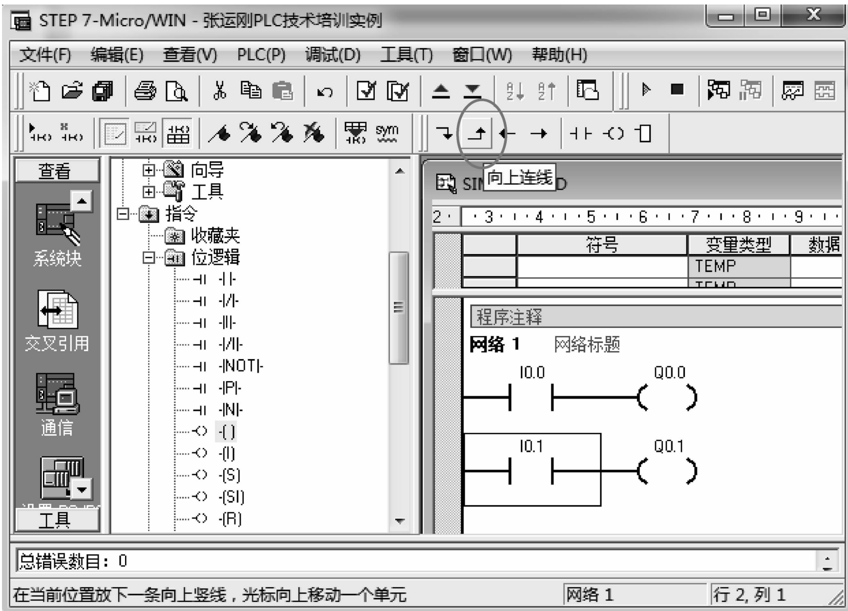


图 1-25　画垂直线

画水平线。如果要求输入如图 1-26 所示的程序，可以先按照输入指令的办法输入如图 1-27 所示的程序，然后把把光标放在 I0. 3 的常开触点指令旁边的“A”，用鼠标单击“→”符号即可画好水平线。

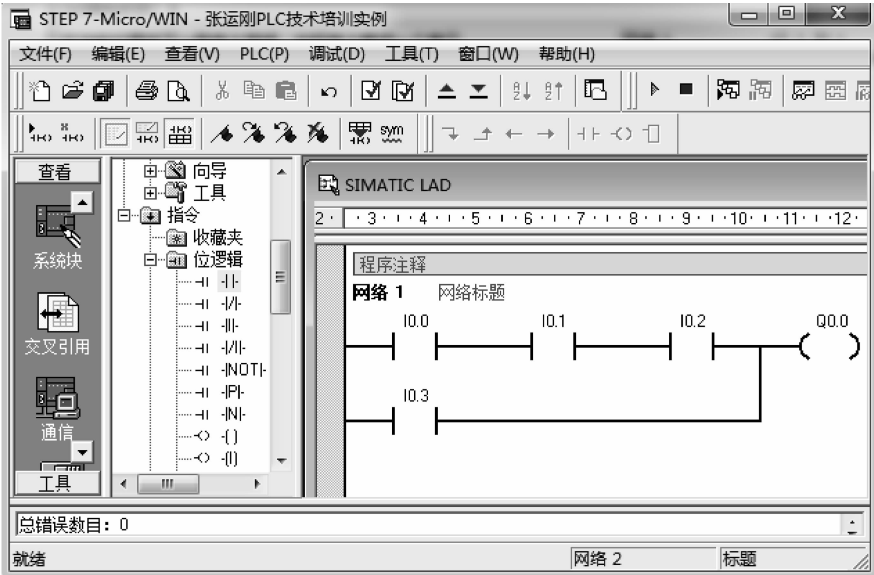


图 1-26　画水平线目标程序

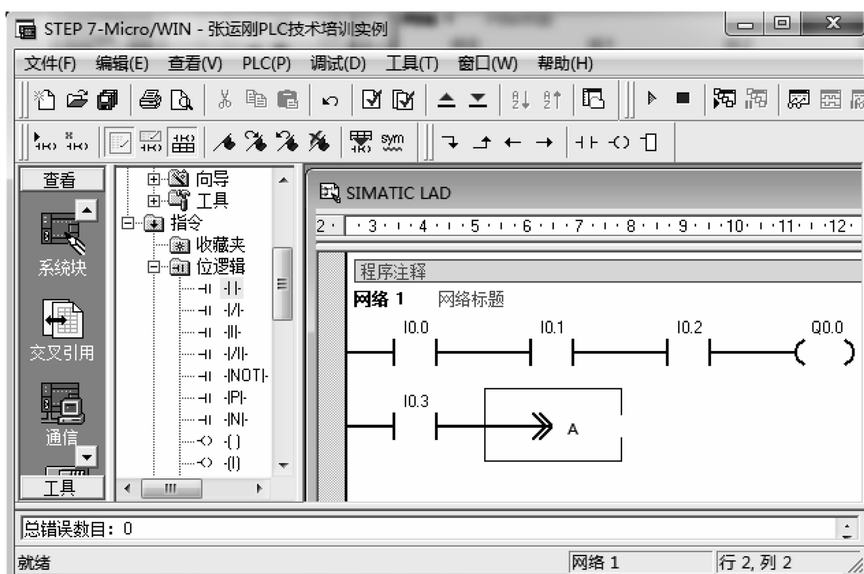


图 1-27　画水平线

问 5：在编程时，怎样插入列和行?

答：插入列。如果要求把图 1-28 所示的程序，编辑为图 1-29 所示的程序，可以把光标放在 I0. 0 的常开触点指令上面，如图 1-30 所示，然后单击“编辑”→“插入”→“列”，如图 1-31 所示，就可以在 I0. 0 前面增加一列的位置，如图 1-32 所示。然后在图 1-32 所示的光标位置放置 M0. 6 的常开触点指令。

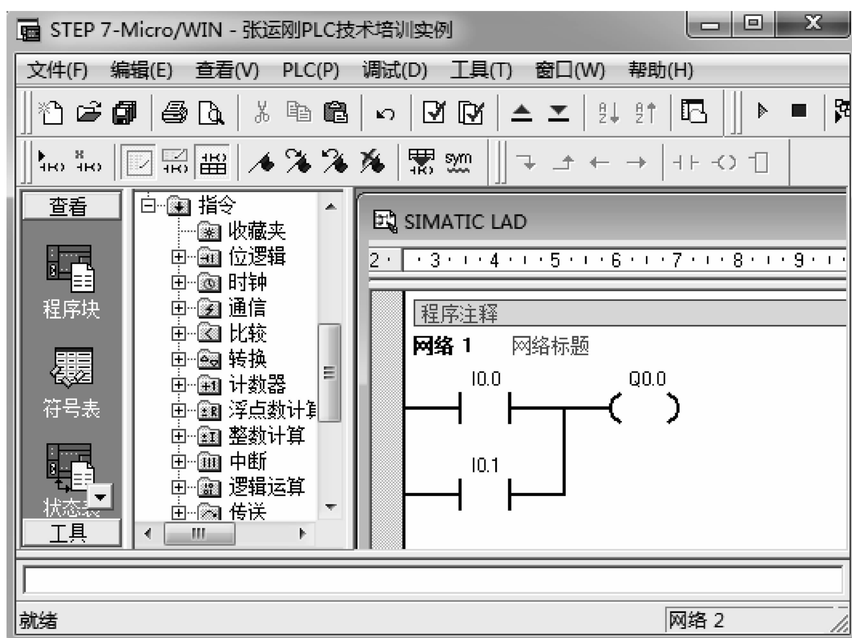


图 1-28　原有程序

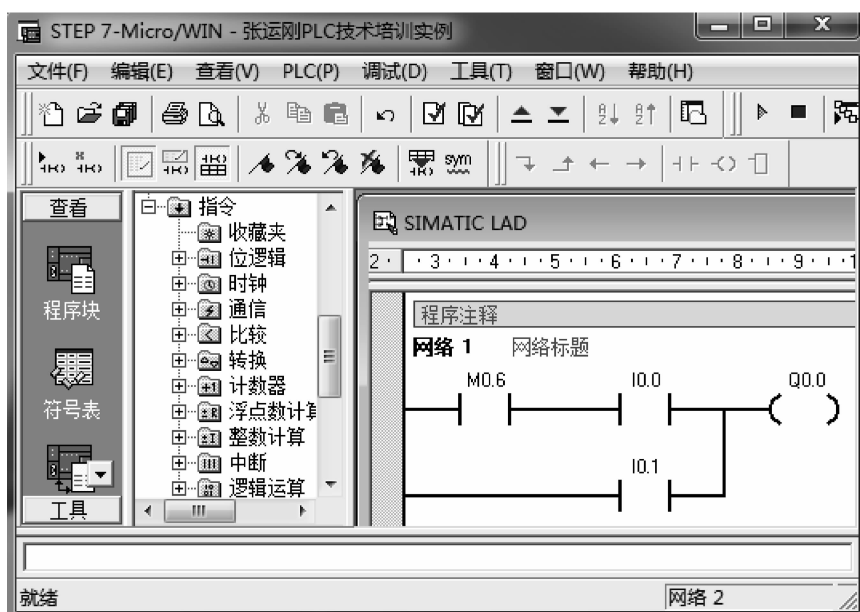


图 1-29　更改后的程序

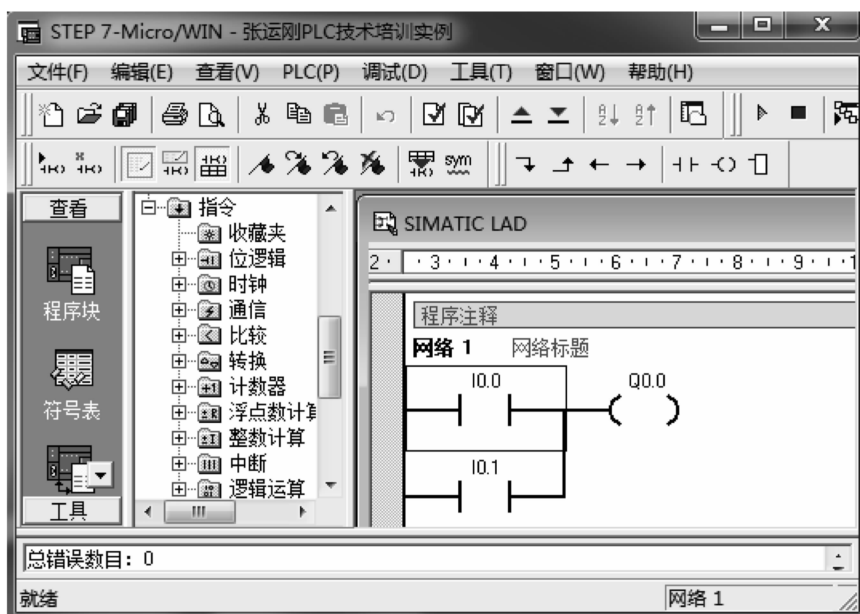


图 1-30　放置光标



图 1-31 插入列

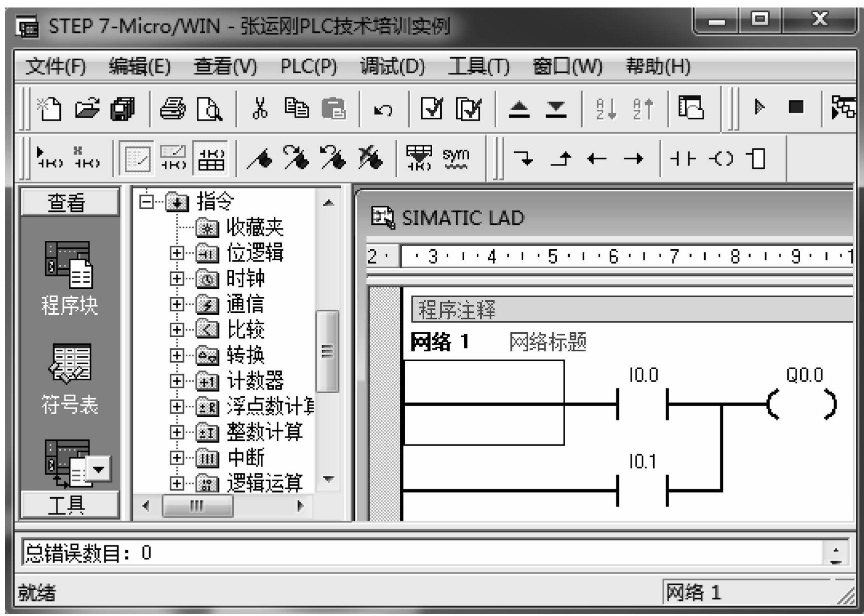


图 1-32 插入一列后的程序

插入行。如果要求把图 1-33 所示的程序，编辑为图 1-34 所示的程序，可以把光标放在 I0.0 的常开触点指令上面，如图 1-35 所示，然后单击“编辑”→“插入”→“行”，如图 1-36 所示，就可以在 I0.0 前面增加一列的位置，如图 1-37 所示。然后在图 1-37 所示的光标位置并联放置 M0.7 的常开触点指令。

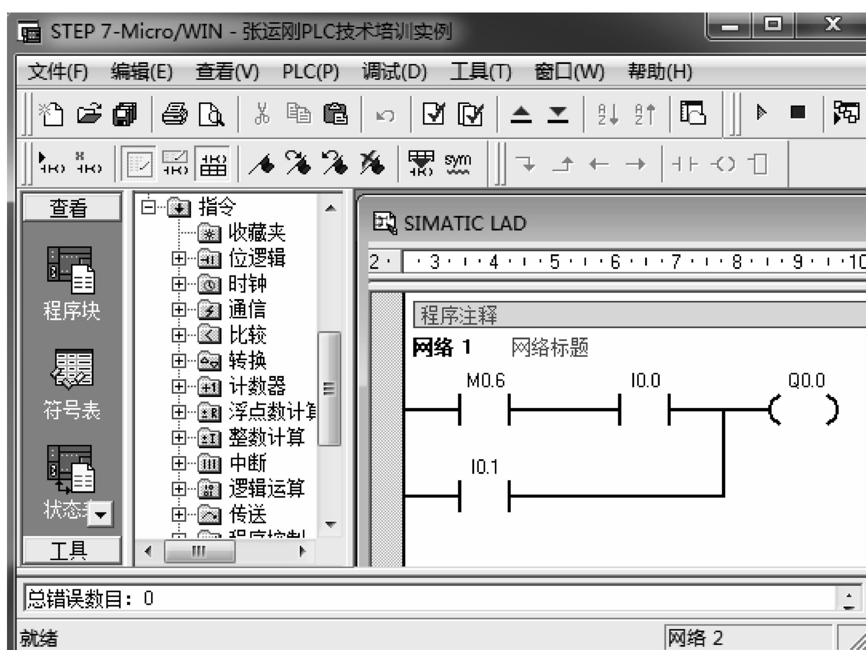


图 1-33　原来程序

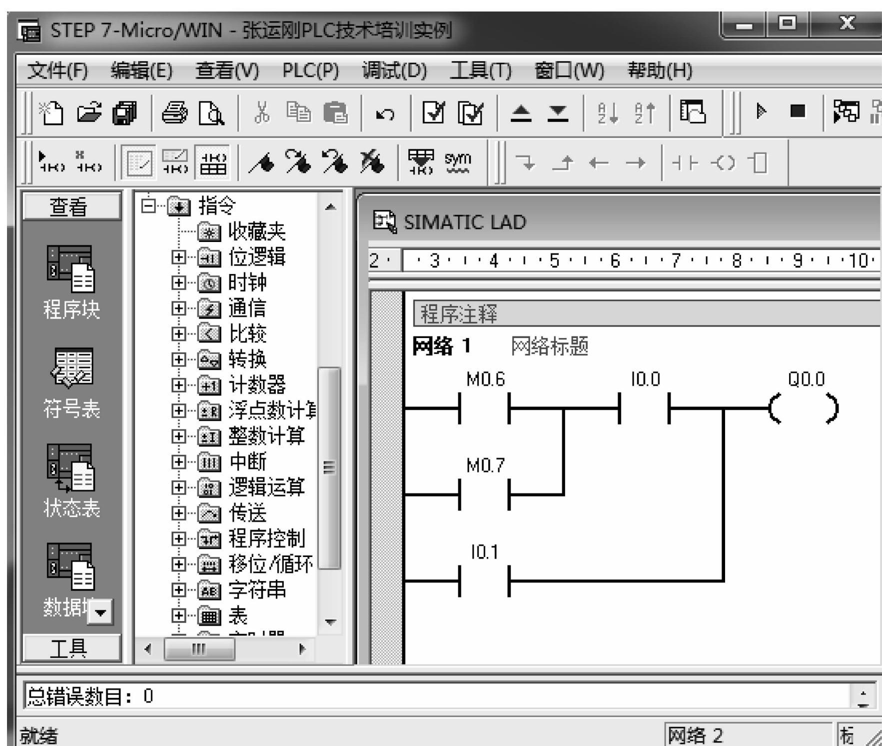


图 1-34　更改后的程序

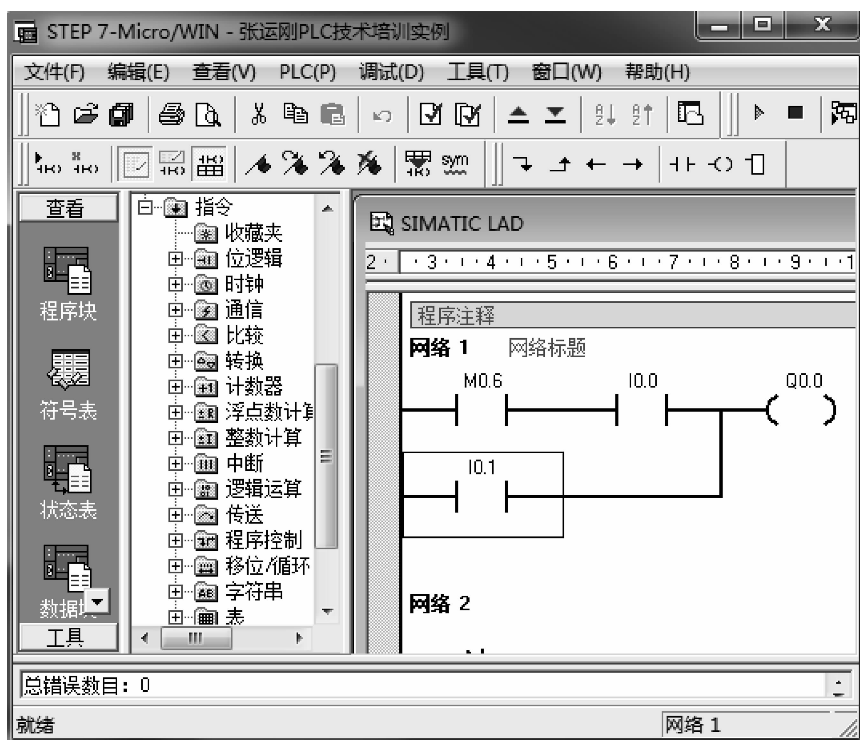


图 1-35　放置光标



图 1-36　插入行

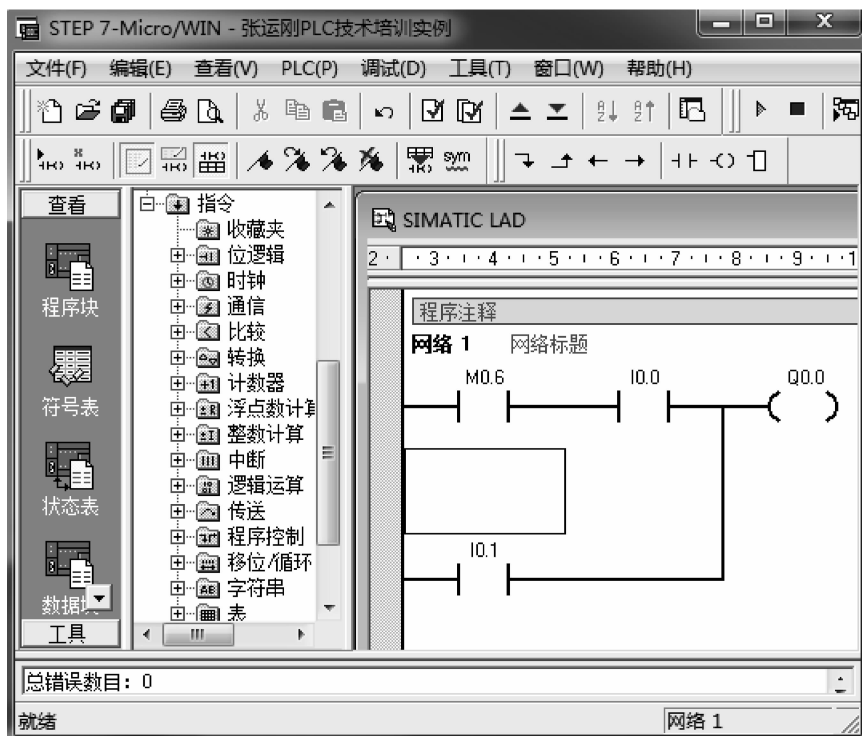


图 1-37　插入一行后的程序

问 6：在编程时，怎样插入和删除网络？

答：插入网络。在编写程序的时候，经常需要在某个地方插入一个网络程序，或者删除一个网络程序。如在图 1-38 所示程序中，需要在网络 1 的前面增加一个网络程序，可以先插入一个网络，把鼠标指向网络 1 的空白位置，然后单击“编辑”→“插入”→“网络”，即可以插入一个网络如图 1-39 所示。

删除网络。如果需要删除一个网络，可以把鼠标指向准备删除网络的空白地方，然后单击“编辑”→“删除”→“网络”，即可以删除当前一个网络，如图 1-40 和图 1-41 所示。

问 7：在编程时，怎样添加和删除程序？

答：在 S7-200 的程序编辑画面中，默认只有 OB1、SBR _0 和 INT _0 3 个程序，OB1 是主程序，SBR _0 是子程序 0，INT _0 是中断程序 0。

如果需要增加子程序，可以用鼠标单击项目栏中的程序块，然后按动鼠标右键，在弹出的下拉菜单中单击“插入”→“子例行程序”，就可以增加一个子程序。

如果需要增加中断程序，可以用鼠标单击项目栏中的程序块，然后按动鼠标右键，在弹出的下拉菜单中单击“插入”→“中断”，就可以增加一个中断程序。

如果需要删除子程序，可以用鼠标选中项目栏程序块里需要删除的子程序，然后按动鼠标右键，在弹出的下拉菜单中单击“删除”，将弹出一个确认对话框，单击“是”确认，即可删除选中的子程序。

如果需要删除中断程序，可以用鼠标选中项目栏程序块里需要删除的中断程序，然后按动鼠标右键，在弹出的下拉菜单中单击“删除”，将弹出一个确认对话框，单击“是”确认，即可删除。

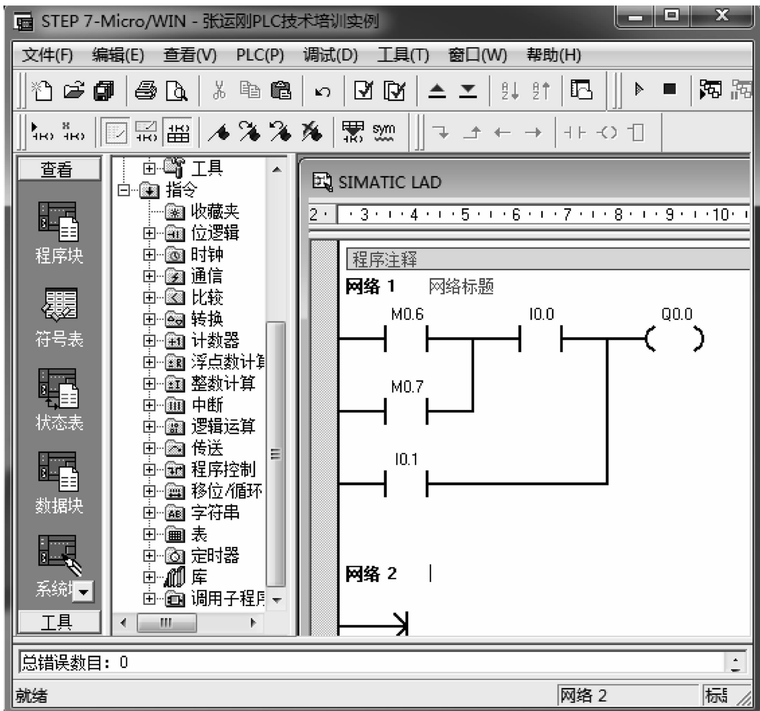


图 1-38　插入网络前

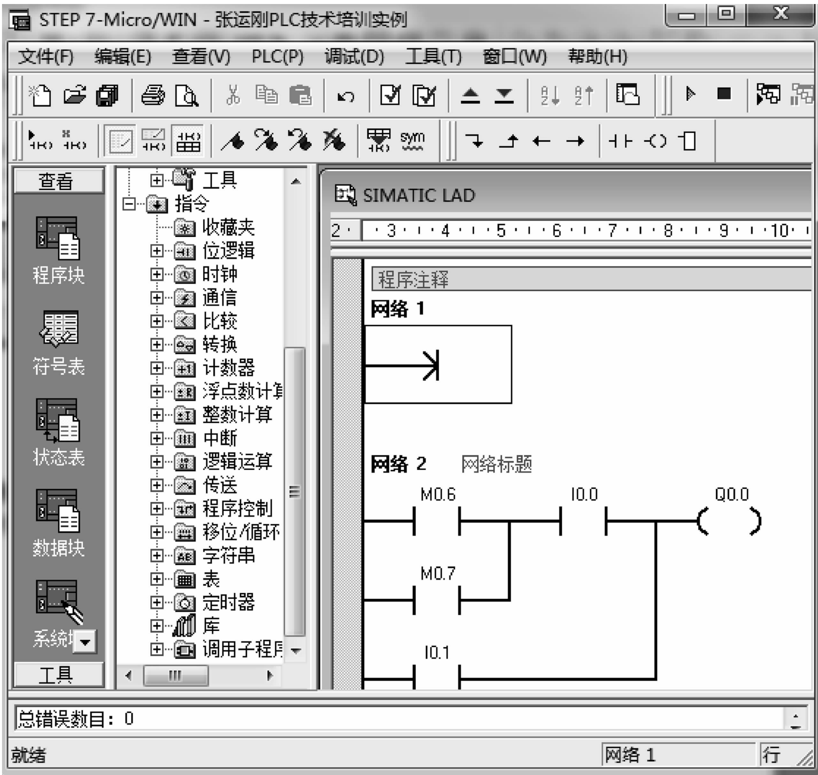


图 1-39　插入网络后

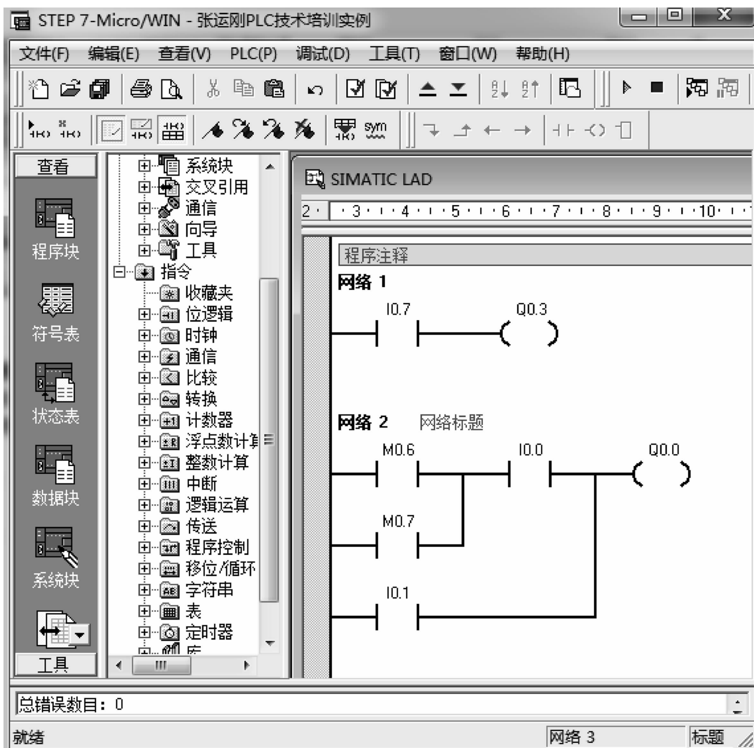


图 1-40　删除网络前

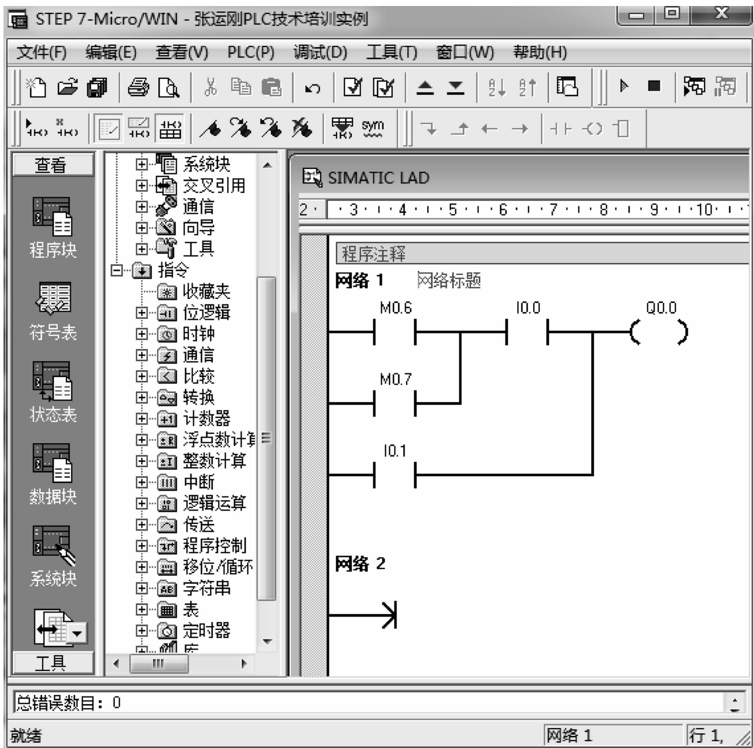


图 1-41　删除网络后

1.3　通信和监控

问：V4.0 STEP 7 MicroWIN SP9 编程软件怎样才能与 CPU 通信？
怎样下载程序到 CPU？
如何进入程序监控状态？
CPU 面板上有一个开关，旁边有 RUN/TERM/STOP 字样，分别是什么意思？

问 1：V4.0 STEP 7 MicroWIN SP9 编程软件怎样才能与 CPU 通信？

答： V4.0 STEP 7 MicroWIN SP9 编程软件与 CPU 通信途径和方式有很多种，其中最常用的是 PC/PPI 方式以及 901-3DB30-0XA0 通信电缆与 CPU 的 PORT0 或者 PORT1 口连接实现通信。具体操作方法是：在编程界面，双击“项目”→“通信”→“通信”字样，如图 1-42 所示，打开通信设置和检测画面，如图 1-43 所示。

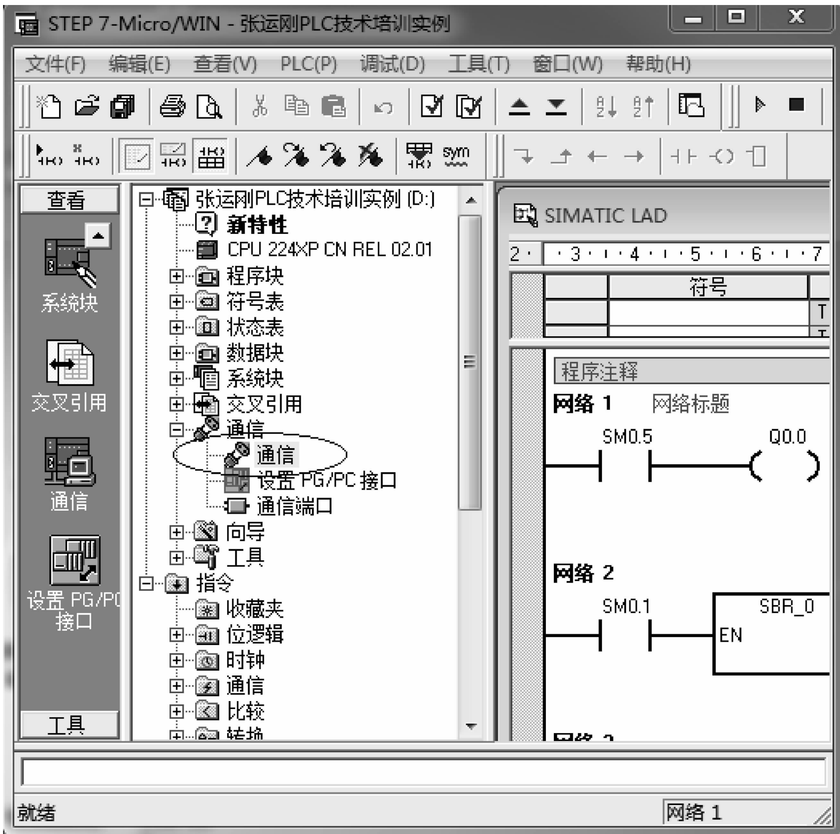


图 1-42　打开通信

在通信设置与检测画面里，单击左下角的“设置 PG/PC 接口”，打开设置 PG/PC 接口界面，如图 1-44 所示。在设置 PG/PC 接口界面中，选择设置 PG/PC 接口类型为“PC/PPI cable PPI. 1”。

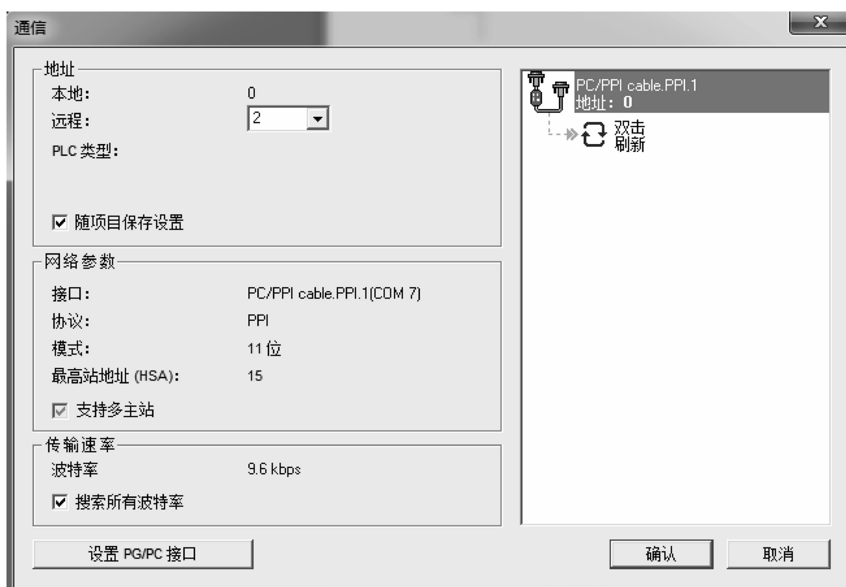


图 1-43　通信设置和检测界面

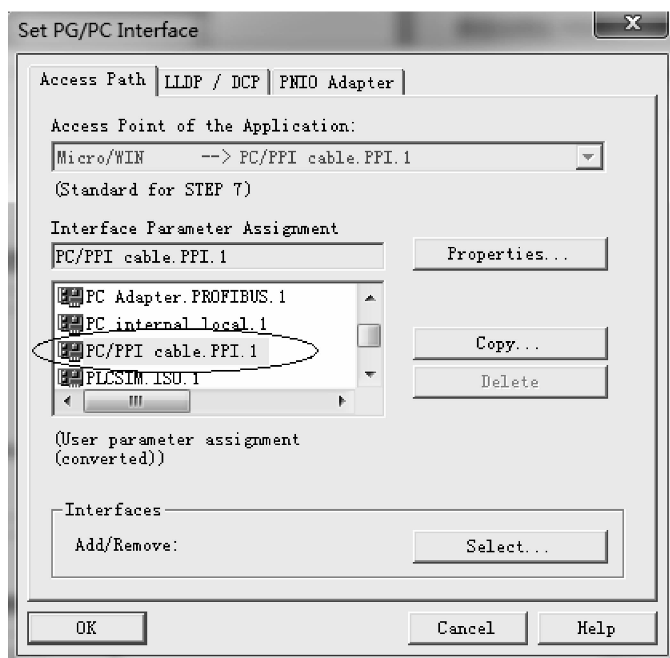


图 1-44　设置 PG/PC 接口

单击图 1-44 的“Properties”按钮打开设置 PG/PC 接口属性画面，在接口属性单击“本地连接”，本地接口一般指编程设备（如编程电脑）通信端口，如图 1-45 所示选择 USB（本例使用 901-3DB30-0XA0 通信电缆）。再单击属性界面中的“PPI”，设置 PPI 通信波特率等，然后单击“OK”按钮。

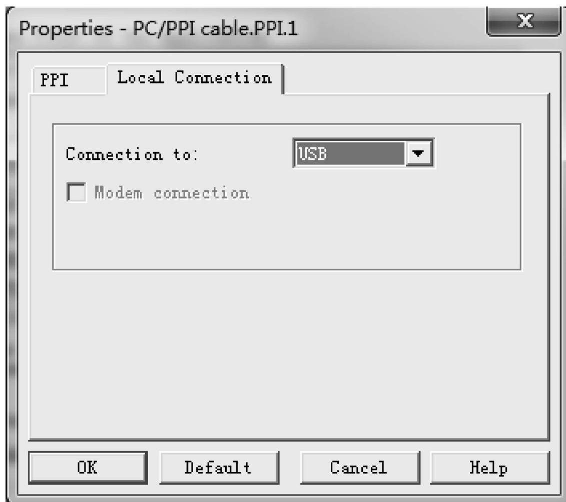


图 1-45　设置 PC/PPI 通信接口

设置完毕 PG/PC 接口属性，使用 PC/PPI 通信电缆将 CPU 与编程设备的通信口连接，单击与 CPU 通信检测界面中的“双击刷新”，自动搜索 PPI 网络上的 CPU 站点，如图 1-46 所示，搜索出来的 CPU 站号、通信波特率、CPU 型号和版本都会显示出来。

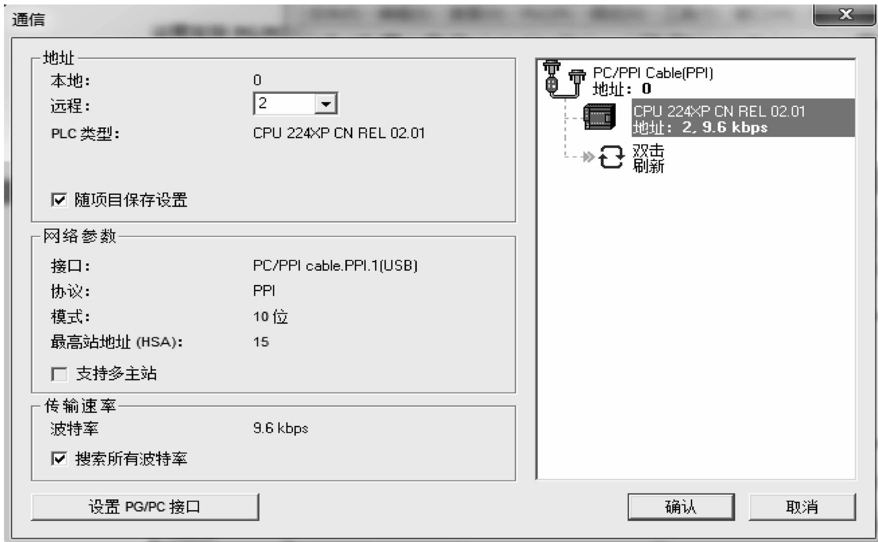


图 1-46　自动搜索站点

问 2：怎样下载程序到 CPU？

答： 按照以下操作很容易实现下载程序到 CPU。单击工具栏的下载符号“▼”，就可以弹出下载界面，如图 1-47 所示，单击“下载”按钮即可下载。

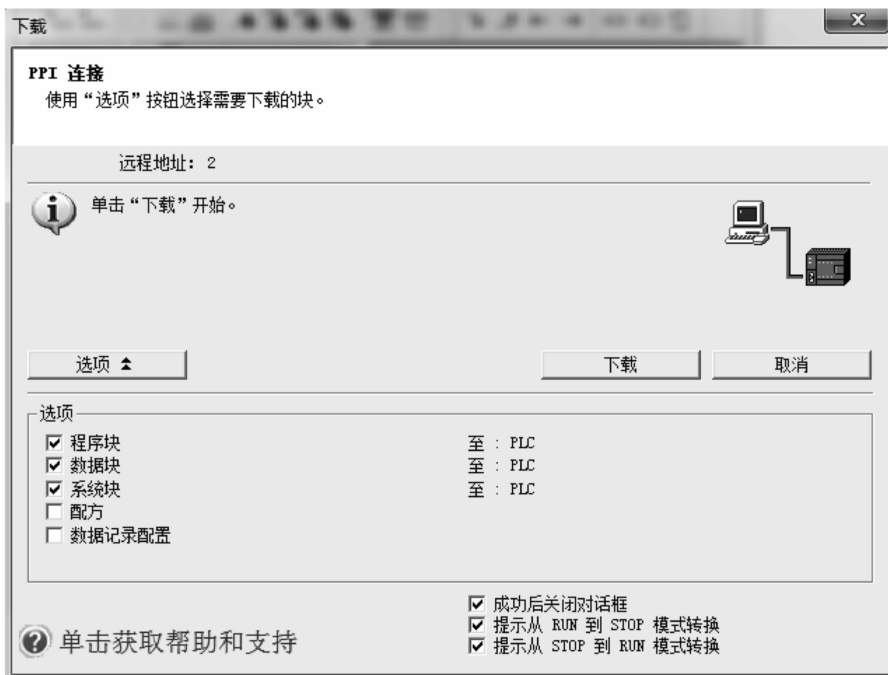


图 1-47　下载项目

问 3：如何进入程序监控状态？

答：程序监控状态有两种：使用执行状态和不使用执行程序状态，这两种状态可以来回切换，都是为了方便监控状态。默认状态是使用执行状态。

在图 1-48 中，单击“调试”→“开始程序状态监控（P）”，即进入执行状态监控程序界面，如图 1-49 所示。如果进入程序监控状态之前，把图 1-48 所示“使用执行状态”前面的“✓”去掉，再单击“调试”→“开始程序状态监控”，即进入不执行状态监控程序界面，如图 1-50 所示。

问 4：CPU 面板上有一个开关，旁边有 RUN/TERM/STOP 字样，分别是什么意思？

答：CPU 面板上有一个模式切换开关，其有 3 个（RUN/TERM/STOP）挡位，当要运行程序时，把模式开关切换到“RUN”位置，观察 CPU 上的 RUN 状态指示灯亮起来，说明在运行程序状态；当要停止运行程序时，把模式开关切换到“STOP”位置，观察 CPU 上的 RUN 状态指示灯熄灭了，说明 CPU 在停止运行程序状态。

当把 CPU 上的 RUN/TERM/STOP 开关扳动到“TERM”位置，我们称为测试挡位。在测试挡位可以通过软件实现远程控制 CPU 的工作模式。比如单击“▶”符号，自动弹出确认运行画面，如图 1-51 所示，单击“是”确认运行，CPU 开始运行用户程序，查看 CPU 上的 RUN 状态指示灯就亮起来了。

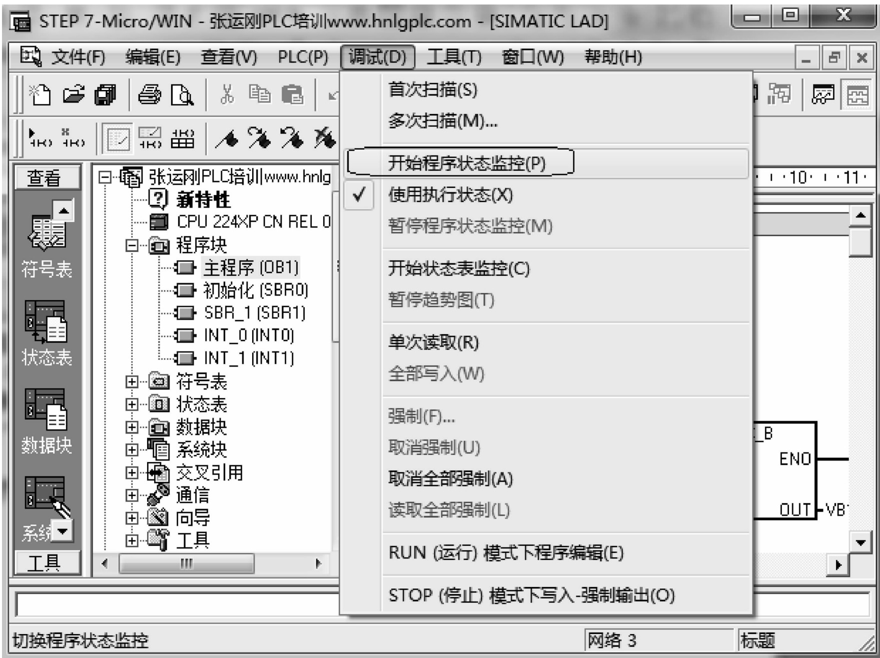


图 1-48　打开程序监控界面

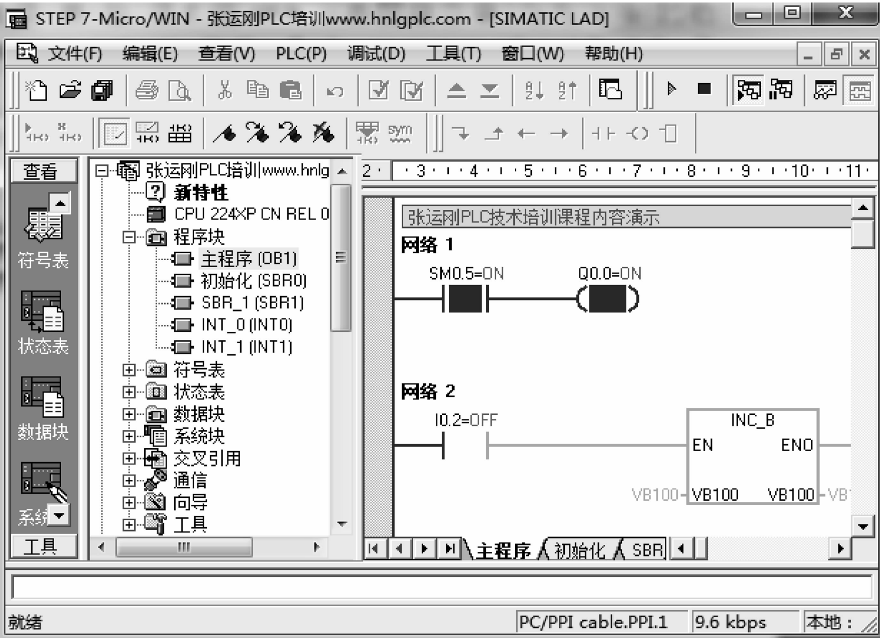


图 1-49　执行状态监控界面

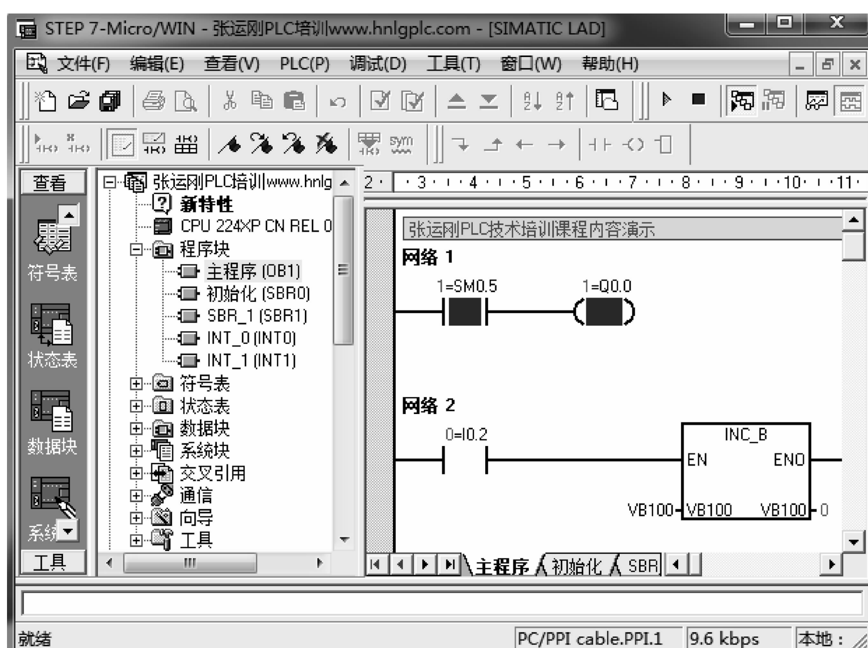


图 1-50　不执行状态监控界面

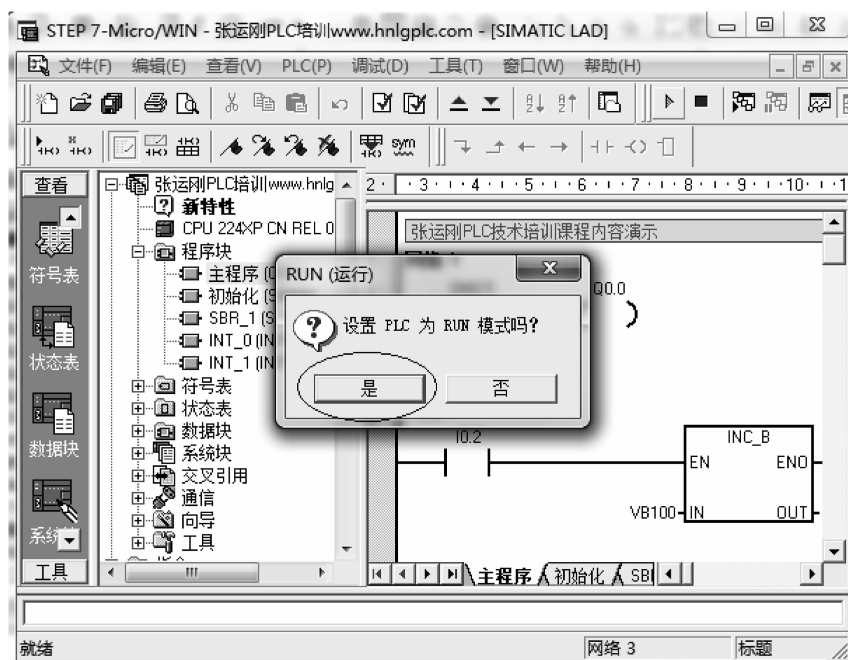


图 1-51　运行用户程序

当把 CPU 上的 RUN/TERM/STOP 开关扳到“TERM”位置上，单击“■”符号，自动弹出确认停止运行画面，单击“是”确认停止运行，查看 CPU 上的 STOP 状态指示灯就亮起来了。

1.4　程序的修改和错误处理

问：在编写程序时，如果发现指令错了，怎样更改指令？
在编写程序时，如果发现软元件错了，怎样更改软元件？
在编写程序过程中，出现编译错误如何处理？
下载程序时，出现错误如何处理？

问1：在编写程序时，如果发现指令错了，怎样更改指令？

答：更改指令一般有两种办法：一是在程序界面直接更改，二是使用插入/覆盖功能修改。

在程序界面直接更改。比如需要把图1-52所示的程序更改为图1-53所示的程序，更改方法是首先删除M0.6常开触点，然后马上放置常闭触点输入软元件M0.6即可。具体操作方法是，选中M0.6常开触点，单击键盘的“Delete”删除键，然后把常闭触点放置在这里并从键盘输入软元件名称即可。

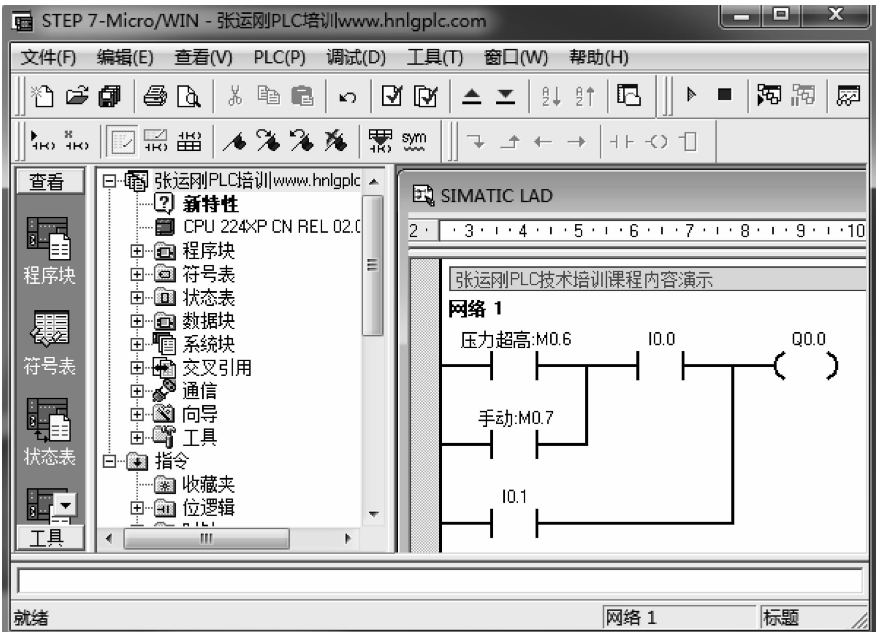


图1-52　原始程序

使用插入/覆盖功能修改。又比如需要把图1-52所示的程序更改为图1-53所示的程序。首先将计算机输入选择为覆盖输入法（OVR），在编程界面中，单击键盘上的“Insert”键，观察编程软件右下角的字样变成“OVR”如图1-54所示，表示已经切换到覆盖输入法了。然后在指令树把常闭触点拉到需要更改的指令上即可，如图1-55所示。

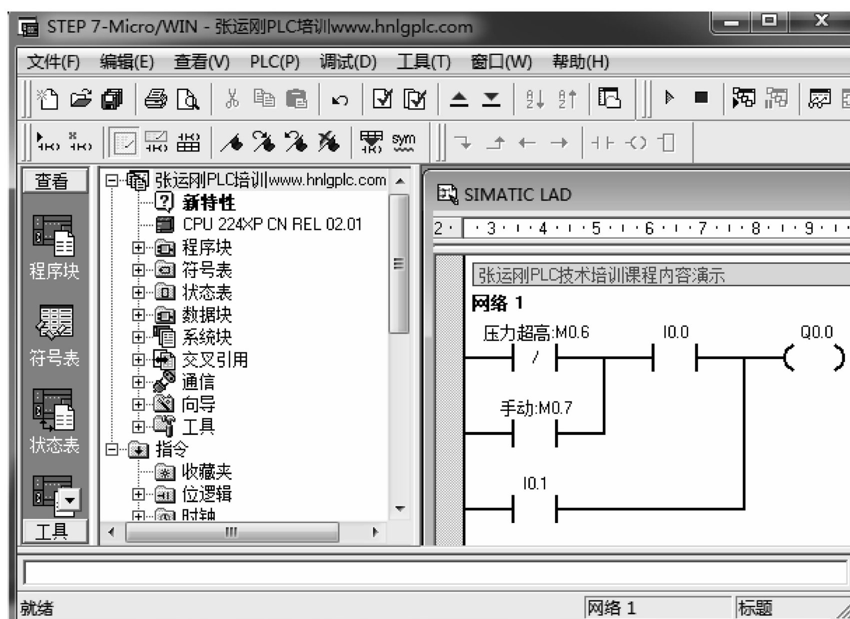


图 1-53　更改后的程序

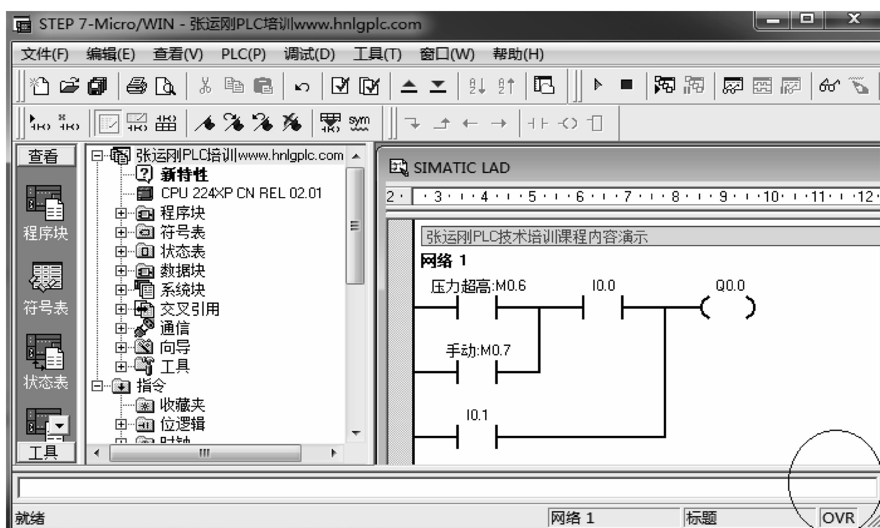


图 1-54　切换到 OVR

问 2：在编写程序时，如果发现软元件错了，怎样更改软元件？

答：更改软元件一般有两种办法，一是直接更改，二是使用编辑功能更改。

直接更改方法简单，在程序界面选中需要更改的软元件直接从键盘输入修改即可。

使用编辑功能更改。比如需要把图 1-56 所示的程序更改为图 1-57 所示的程序。单击文件栏的“编辑”→“替换”，在弹出界面选择“替换”，在查找栏目输入“M0. 6”，如图 1-58 所示，在替换为栏目输入“I0. 3”，然后单击“全部替换”按钮，这样所有 M0. 6 软元件都替换为 I0. 3 了。使用编辑功能合适于多处需要更改，如果人工查找更改就显得麻烦了，一是工作量大，二是容易漏改。

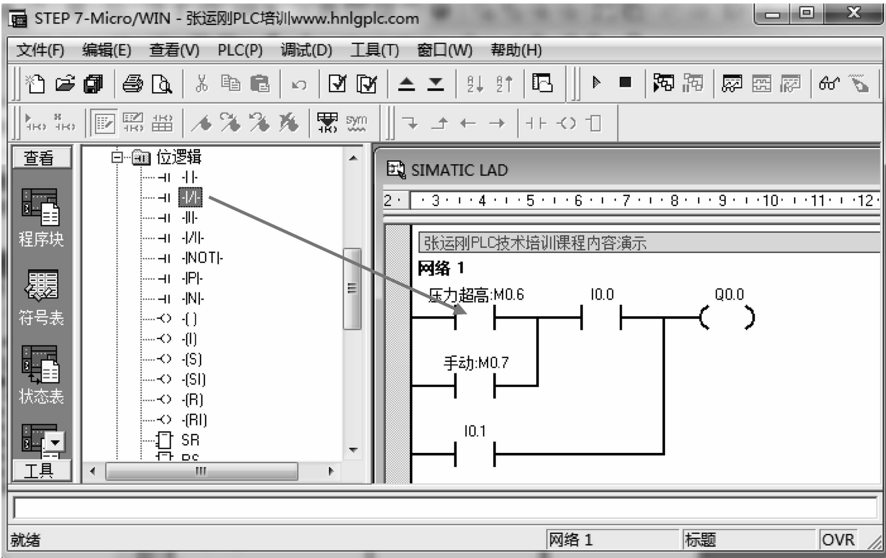


图 1-55　覆盖指令

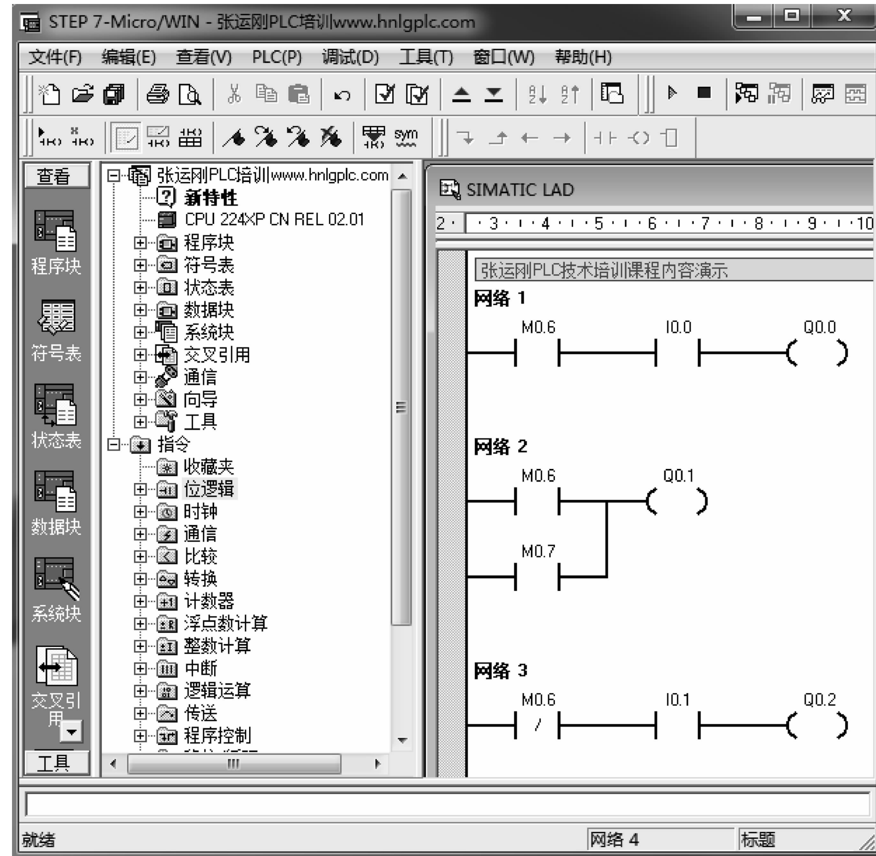


图 1-56　更改前的程序

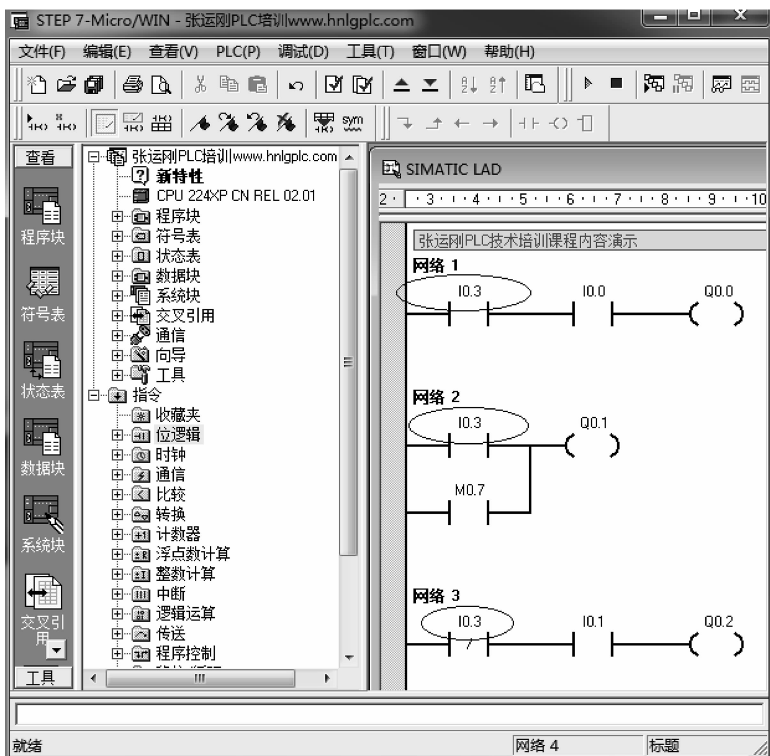


图 1-57　更改后的程序



图 1-58　替换方法

问 3：在编写程序过程中，出现编译错误如何处理？

答：在编程的时候出现编译错误，比如在图 1-59 所示的程序界面中，单击“ ”按钮要求对程序进行编译，将会在状态输出窗口显示编译的状态，如果有错误，双击出错提示的行如“网络 3，行 1，列 2：错误 44：开路。”，光标会转到出错的程序位置，这样查找编译错误非常方便。

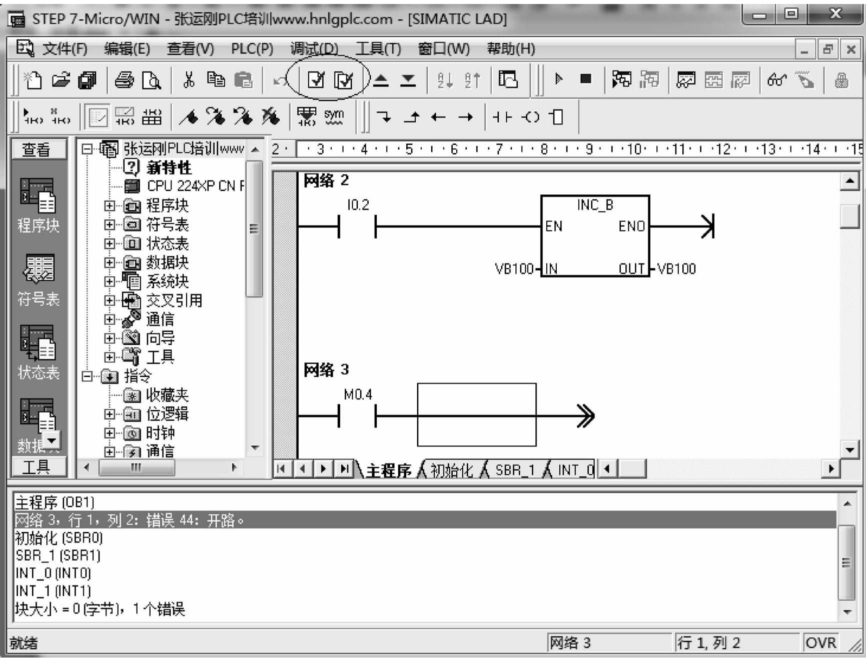


图 1-59　编译错误

问 4：下载程序时，出现错误如何处理？

答：下载程序的时候出现错误，比如在程序界面中，单击“ ”按钮要求对程序进行下载，将会弹出下载操作界面，按照提示进行下载，在下载过程中弹出出错界面，如图 1-60 所示。这时候把出错界面关闭，单击文件栏的“PLC”→“信息”，将会弹出具体错误提示，如图 1-61 所示。按照提示信息相应作出处理即可。


STEP 7-Micro/WIN - 张运刚PLC培训www.hnlgplc.com - [SIMATIC LAD]
文件(F) 编辑(E) 查看(V) PLC(P) 调试(D) 工具(T) 窗口(W) 帮助(H)
查看
程序块
符号表
状态
工具
张运刚PLC培训www
新特性
CPU 224XP CN
程序块
符号表
状态表
数据块
系统块
交叉引用
通信
向导
工具
指令
网络 2
I0.2
10
JMP
网络 3
M0.4
Q0.2
主程序
初始化
下载失败。
就绪
下载
PPI 连接
使用“选项”按钮选择需要下载的块。
远程地址: 2
CPU 224XP CN REL 02.01
出现编译错误，如需更多信息，请查看非致命错误。
选项
取消
程序块
数据块
系统块
配方
数据记录配置
至 : PLC
至 : PLC
至 : PLC
成功后关闭对话框
提示从 RUN 到 STOP 模式转换
提示从 STOP 到 RUN 模式转换
单击获取帮助和支持

图 1-60　下载错误

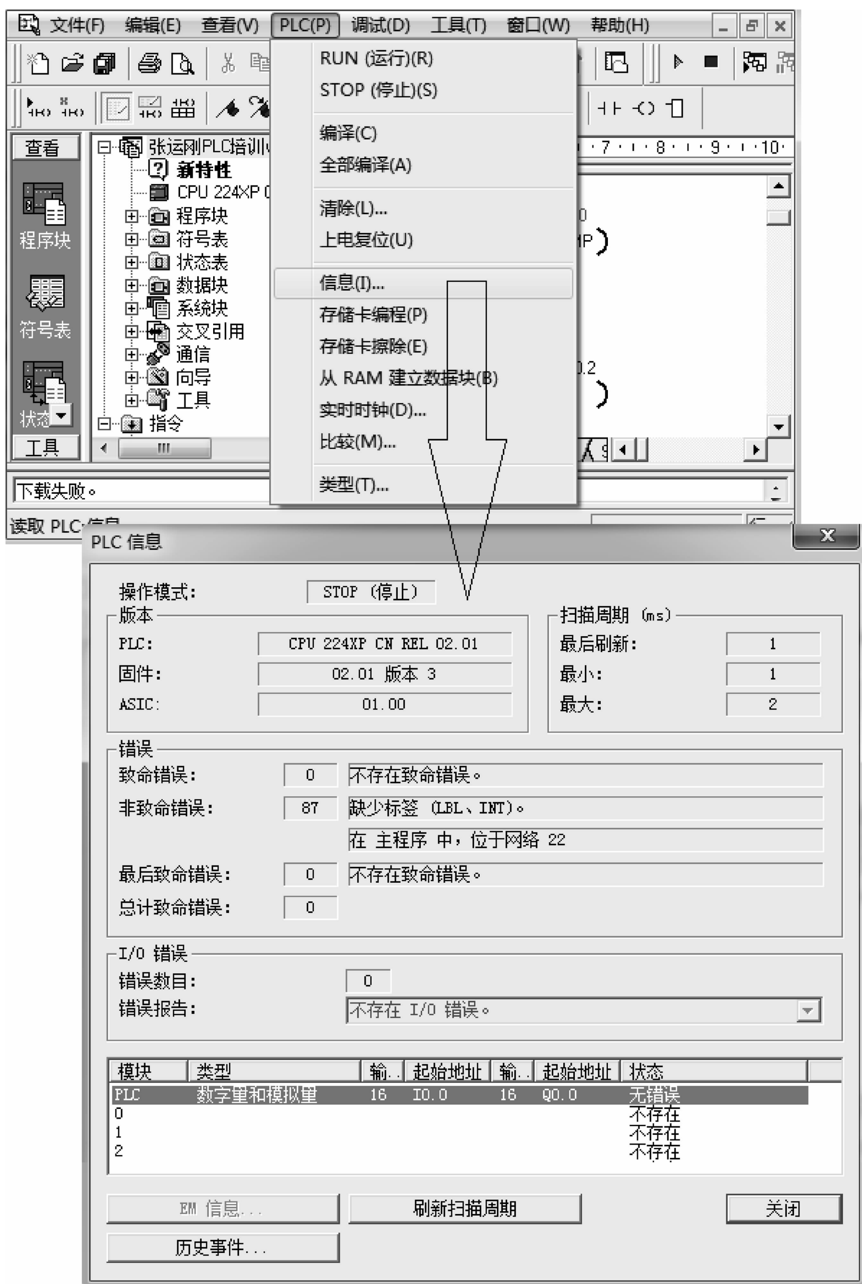


图 1-61　查看 CPU 信息

1.5　程序的注释和项目保存

> 问：编程时使用中文注释可以吗？如何使用？
> 　　项目如何保存在计算机硬盘里？

问 1：编程时使用中文注释可以吗？如何使用？

答：编程时可以使用中文注释，中文注释有软元件符号、软元件注释、POU 符号、POU 注释以及网络标题和网络注释。下面将详细论述如何使用。

定义元件符号注释的方法一般有两种，一种是在程序界面定义，另一种是在符号表里面定义。如果需要修改符号注释，只能在符号表中进行修改。

在编程界面定义元件符号注释。如在图 1-62 的程序中，先把鼠标指向需要定义符号注释的元件 M0.6，然后单击鼠标右键，在弹出的下拉菜单中单击“定义符号”，在弹出的定义符号界面里写上元件的符号注释，如图 1-63 所示，然后单击“确认”即可。

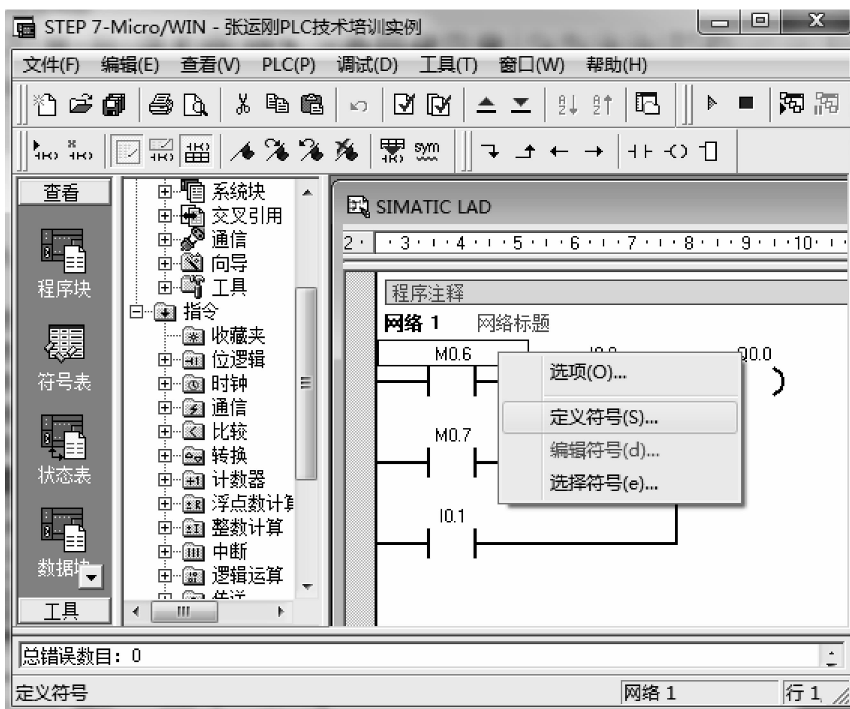


图 1-62　打开定义软元件符号注释

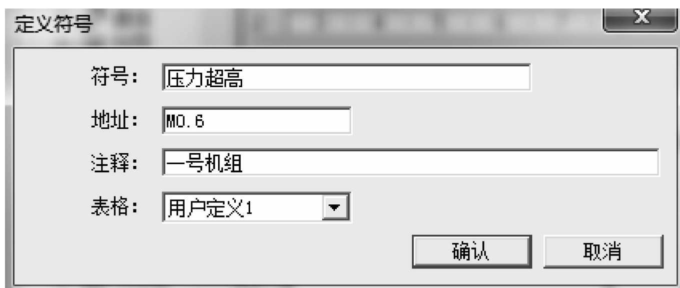


图 1-63　定义元件符号注释

在符号表界面定义元件的符号。如在图 1-64 中单击查看里面的符号表图标，即可打开

符号表定义界面。在符号表定义界面定义元件符号，此种方法适合对大批量软元件进行定义符号注释，如图 1-65 所示。

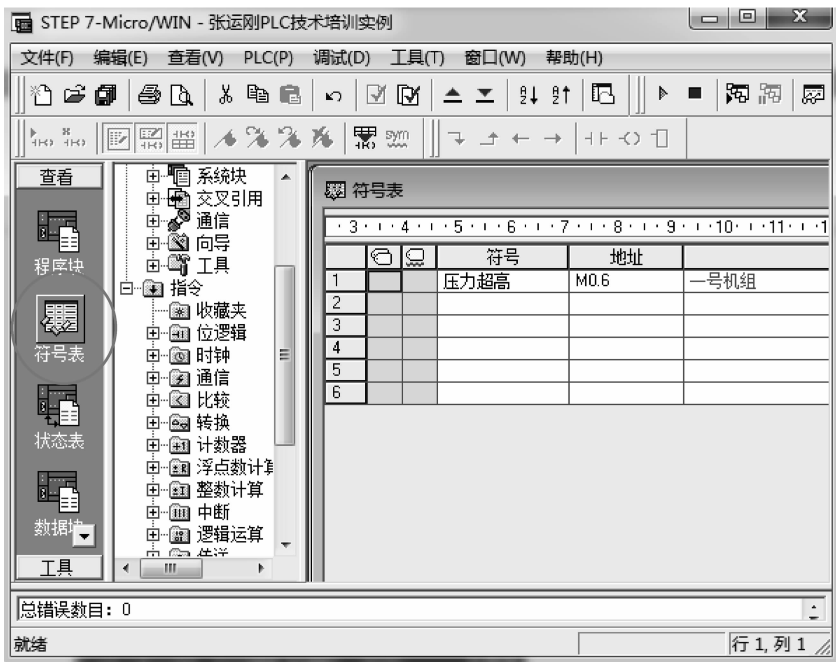


图 1-64　打开符号表编辑器



图 1-65　在符号表界面编辑符号注释

程序界面显示符号和地址。在图 1-66 中单击“工具”→“选项”，打开程序编辑器，在程序编辑器界面里，可以定义指令显示的大小，软元件显示可以选择只显示符号或符号和地址同时显示，还可以选择符号显示的字体和大小等，如图 1-67 所示。



图 1-66　打开选项



图 1-67　程序编辑器

在程序界面显示或隐藏符号注释。如果希望把软元件的符号显示在程序界面，单击“查看”→“符号寻址”，如图 1-68 所示，软元件的符号就可以显示在程序界面，如图 1-69 所示。

如果不希望把元件的符号显示在程序界面，再次单击“查看”→“符号寻址”即可，软元件的符号就在程序界面隐藏了。

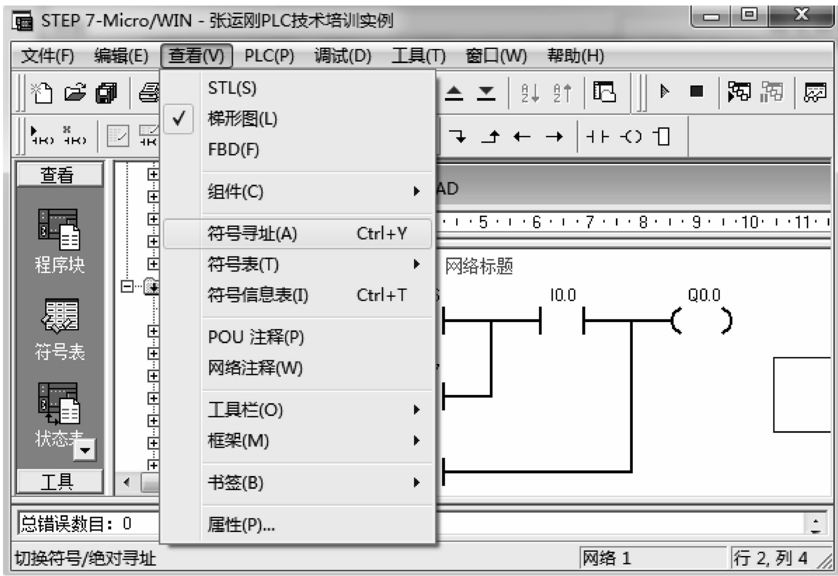


图 1-68　选择在程序界面显示元件符号

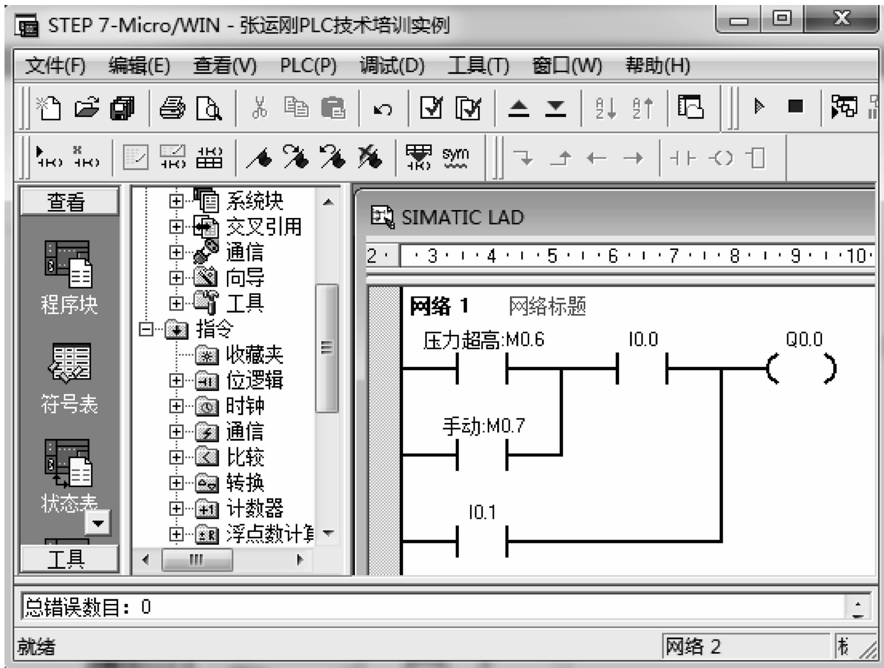


图 1-69　程序界面显示软元件符号

在程序界面显示或隐藏元件符号信息表。如果希望把元件的符号信息表显示在程序界面，单击“查看”→“符号信息表”，软元件的符号信息表就可以显示在程序界面，如图 1-70 所示。

如果不希望把元件的符号信息表显示在程序界面，再次单击“查看”→“符号信息表”即可，符号信息表就可以隐藏在程序界面了。

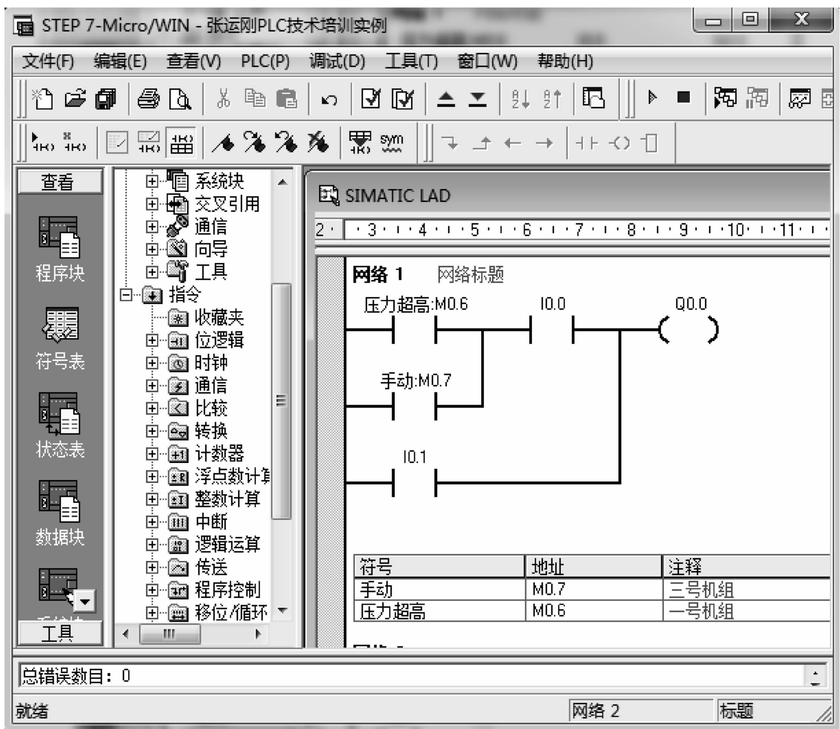


图 1-70　在程序界面显示符号信息表

POU 符号注释。在编写程序时，不但软元件可以使用符号注释，POU（程序）也可以使用符号注释。查看 POU 符号注释的方法是，单击项目树→“符号表”→“POU 符号”，将会弹出 POU 符号表，如图 1-71 所示。每添加一个程序，系统会自动分配符号和注释。

如果有需要允许用户更改 POU 符号，方法是展开项目树下面“程序块”，在程序块下面显示现有的程序符号和地址，右键单击需要更改名称的程序，在弹出对话框中选择“重命名”如图 1-72 所示，将会出现重命名窗口，在该窗口输入新符号即可。



图 1-71　查看 POU 符号注释



图 1-72　程序重命名

如果有需要也允许用户更改 POU 注释，方法是展开项目树下面“程序块”，在程序块下面显示现有的程序符号和地址，双击需要更改注视的程序块，将会打开该程序如图 1-73 所示，在图 1-73 的程序注释地方更改为新注释即可。

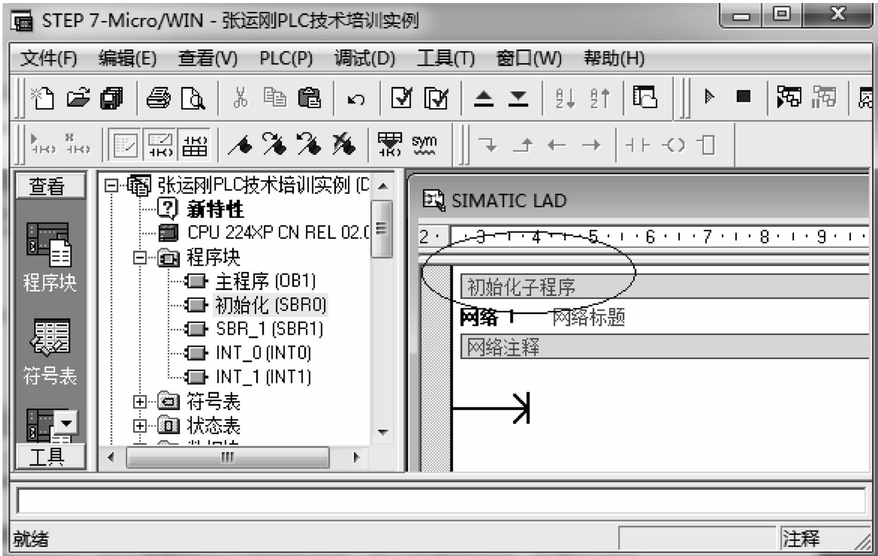


图 1-73　更改程序注释

网络标题和网络注释。每一个网络，系统会自动定义网络标题和网络注释，只不过这些标题和注释都是相同的。如果有需要允许用户更改网络标题和网络注释，方法是把原来网络标题和网络注释字样更改即可，如图 1-74 所示。

问 2：项目如何保存在计算机硬盘里？

答：在 STEP 7-MicroWIN 编程软件的编程界面，单击“文件”→“保存”或者“另存为”，如图 1-75 所示，在出现的画面中选择保存项目文件的路径并写上项目文件的名称，如图 1-76 所示。如果选择另存为，而且另存为原来已经存在的项目名称，会弹出要求覆盖确

认对话框，如果需要覆盖单击“是”按钮，如图 1-77 所示。

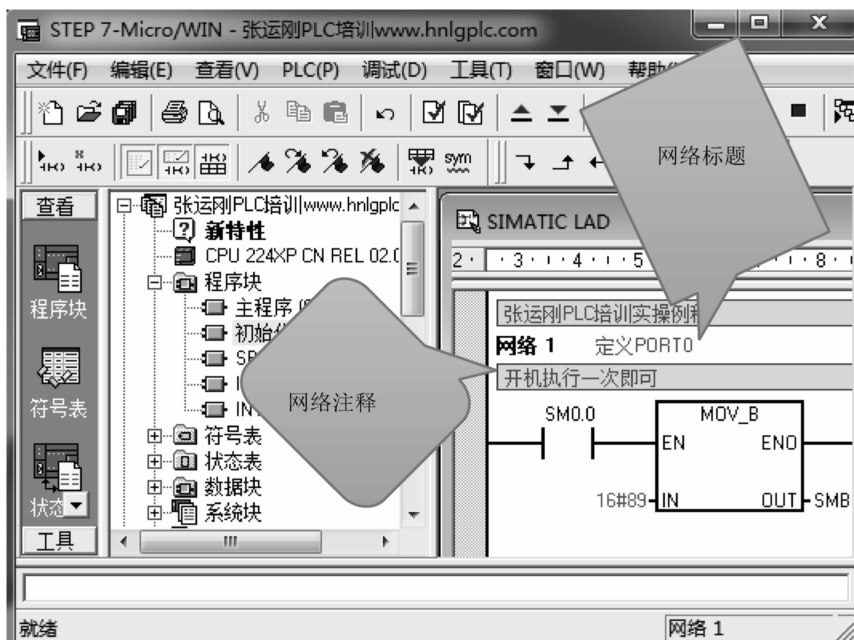


图 1-74 更改网络标题和注释



图 1-75 保存项目



图 1-76　项目保存路径和名称

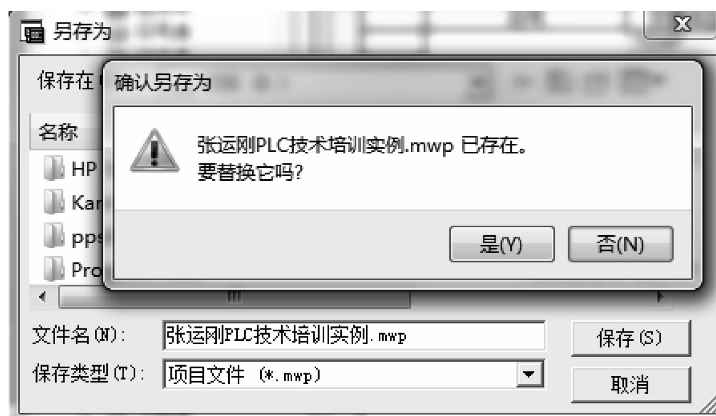


图 1-77　确认覆盖替换

第 2 章　S7-200 的软元件

2.1　I/Q 输入/输出

> 问：在程序中看到有 I 和 Q 的符号，如何理解 I 和 Q？

问：在程序中看到有 I 和 Q 的符号，如何理解 I 和 Q？

答：在程序中的 I 是输入映像存储器，Q 是输出映像存储器。可以从其功能、寻址范围和访问长度几方面理解它。

I 输入是 PLC 从外部开关接收信号的窗口。可以理解为在 PLC 内部与 PLC 的输入端子相连的输入映像存储器（I）是一种光绝缘的电子继电器，它有无数的常开触点（—| |—）与常闭触点（—|/|—），这些常开、常闭触点可随意使用。I 可以用程序驱动，但有物理点的 I 的状态又取决于外部的物理开关状态；没有物理点对应的 I，当驱动该线圈时，其相应的触点会动作：常开触点闭合，常闭触点断开。

在每一个扫描周期的开始，CPU 对物理输入点进行采样，并将采样值传输到相应的输入映像存储器中。

S7-200 系列 PLC，输入映像存储器的“I”可以按位、字节、字或双字来使用。

当按位使用时，地址编号范围是 I0.0 ~ I15.7，共 128 个位；

当按字节使用时，地址编号范围是 IB0 ~ IB15，共 16 个字节；

当按字使用时，地址编号范围是 IW0 ~ IW14，共 8 个字；

当按双字使用时，地址编号范围是 ID0 ~ ID12，共 4 个双字。

输入映像存储器位、字节、字与双字之间的关系见表 2-1。

Q 输出是 PLC 向外部负载发送控制信号的窗口。输出映像存储器 Q 的外部输出触点在 PLC 内与该输出端子相连。Q 有无数的常开触点（—| |—）与常闭触点（—|/|—），这些常开、常闭触点在 PLC 内部可随意使用。当驱动 Q 线圈时，其相应的触点会动作：常开触点闭合，常闭触点断开，有物理点输出的 Q，其输出接点会接通可以驱动外面负载。

在每一个扫描周期的最后，CPU 将输出映像存储器的状态成批地传送到相应的物理输出点上。

S7-200 系列 PLC，输出映像存储器的“Q”可以按位、字节、字或双字来使用。

当按位使用时，地址编号范围是 Q0.0 ~ Q15.7，共 128 个位；

当按字节使用时，地址编号范围是 QB0 ~ QB15，共 16 个字节；

当按字使用时，地址编号范围是 QW0 ~ QW14，共 8 个字；

当按双字使用时，地址编号范围是 QD0 ~ QD12，共 4 个双字。

表 2-1　输入映像存储器位、字节、字与双字之间的关系表

<table>
<tr><th colspan="4">双字</th><th colspan="2">字</th><th>字节</th><th colspan="8">位</th></tr>
<tr><td></td><td></td><td></td><td rowspan="4">ID0</td><td></td><td rowspan="2">IW0</td><td>IB0</td><td>I0. 7</td><td>I0. 6</td><td>I0. 5</td><td>I0. 4</td><td>I0. 3</td><td>I0. 2</td><td>I0. 1</td><td>I0. 0</td></tr>
<tr><td></td><td></td><td rowspan="4">ID1</td><td rowspan="2">IW1</td><td>IB1</td><td>I1. 7</td><td>I1. 6</td><td>I1. 5</td><td>I1. 4</td><td>I1. 3</td><td>I1. 2</td><td>I1. 1</td><td>I1. 0</td></tr>
<tr><td></td><td rowspan="4">ID2</td><td rowspan="2">IW2</td><td>IB2</td><td>I2. 7</td><td>I2. 6</td><td>I2. 5</td><td>I2. 4</td><td>I2. 3</td><td>I2. 2</td><td>I2. 1</td><td>I2. 0</td></tr>
<tr><td rowspan="4">ID3</td><td rowspan="2">IW3</td><td>IB3</td><td>I3. 7</td><td>I3. 6</td><td>I3. 5</td><td>I3. 4</td><td>I3. 3</td><td>I3. 2</td><td>I3. 1</td><td>I3. 0</td></tr>
<tr><td rowspan="4">ID4</td><td rowspan="2">IW4</td><td>IB4</td><td>I4. 7</td><td>I4. 6</td><td>I4. 5</td><td>I4. 4</td><td>I4. 3</td><td>I4. 2</td><td>I4. 1</td><td>I4. 0</td></tr>
<tr><td rowspan="4">ID5</td><td rowspan="2">IW5</td><td>IB5</td><td>I5. 7</td><td>I5. 6</td><td>I5. 5</td><td>I5. 4</td><td>I5. 3</td><td>I5. 2</td><td>I5. 1</td><td>I5. 0</td></tr>
<tr><td rowspan="4">ID6</td><td rowspan="2">IW6</td><td>IB6</td><td>I6. 7</td><td>I6. 6</td><td>I6. 5</td><td>I6. 4</td><td>I6. 3</td><td>I6. 2</td><td>I6. 1</td><td>I6. 0</td></tr>
<tr><td rowspan="4">ID7</td><td rowspan="2">IW7</td><td>IB7</td><td>I7. 7</td><td>I7. 6</td><td>I7. 5</td><td>I7. 4</td><td>I7. 3</td><td>I7. 2</td><td>I7. 1</td><td>I7. 0</td></tr>
<tr><td rowspan="4">ID8</td><td rowspan="2">IW8</td><td>IB8</td><td>I8. 7</td><td>I8. 6</td><td>I8. 5</td><td>I8. 4</td><td>I8. 3</td><td>I8. 2</td><td>I8. 1</td><td>I8. 0</td></tr>
<tr><td rowspan="4">ID9</td><td rowspan="2">IW9</td><td>IB9</td><td>I9. 7</td><td>I9. 6</td><td>I9. 5</td><td>I9. 4</td><td>I9. 3</td><td>I9. 2</td><td>I9. 1</td><td>I9. 0</td></tr>
<tr><td rowspan="4">ID10</td><td rowspan="2">IW10</td><td>IB10</td><td>I10. 7</td><td>I10. 6</td><td>I10. 5</td><td>I10. 4</td><td>I10. 3</td><td>I10. 2</td><td>I10. 1</td><td>I10. 0</td></tr>
<tr><td rowspan="4">ID11</td><td rowspan="2">IW11</td><td>IB11</td><td>I11. 7</td><td>I11. 6</td><td>I11. 5</td><td>I11. 4</td><td>I11. 3</td><td>I11. 2</td><td>I11. 1</td><td>I11. 0</td></tr>
<tr><td rowspan="4">ID12</td><td rowspan="2">IW12</td><td>IB12</td><td>I12. 7</td><td>I12. 6</td><td>I12. 5</td><td>I12. 4</td><td>I12. 3</td><td>I12. 2</td><td>I12. 1</td><td>I12. 0</td></tr>
<tr><td></td><td rowspan="2">IW13</td><td>IB13</td><td>I13. 7</td><td>I13. 6</td><td>I13. 5</td><td>I13. 4</td><td>I13. 3</td><td>I13. 2</td><td>I13. 1</td><td>I13. 0</td></tr>
<tr><td></td><td></td><td rowspan="2">IW14</td><td>IB14</td><td>I14. 7</td><td>I14. 6</td><td>I14. 5</td><td>I14. 4</td><td>I14. 3</td><td>I14. 2</td><td>I14. 1</td><td>I14. 0</td></tr>
<tr><td></td><td></td><td></td><td></td><td>IB15</td><td>I15. 7</td><td>I15. 6</td><td>I15. 5</td><td>I15. 4</td><td>I15. 3</td><td>I15. 2</td><td>I15. 1</td><td>I15. 0</td></tr>
</table>

输出映像存储器位、字节、字与双字之间的关系，见表 2-2。

表 2-2　输出映像存储器位、字节、字与双字之间的关系表

<table>
<tr><th colspan="4">双字</th><th colspan="2">字</th><th>字节</th><th colspan="8">位</th></tr>
<tr><td></td><td></td><td></td><td rowspan="4">QD0</td><td></td><td rowspan="2">QW0</td><td>QB0</td><td>Q0. 7</td><td>Q0. 6</td><td>Q0. 5</td><td>Q0. 4</td><td>Q0. 3</td><td>Q0. 2</td><td>Q0. 1</td><td>Q0. 0</td></tr>
<tr><td></td><td></td><td rowspan="4">QD1</td><td rowspan="2">QW1</td><td>QB1</td><td>Q1. 7</td><td>Q1. 6</td><td>Q1. 5</td><td>Q1. 4</td><td>Q1. 3</td><td>Q1. 2</td><td>Q1. 1</td><td>Q1. 0</td></tr>
<tr><td></td><td rowspan="4">QD2</td><td rowspan="2">QW2</td><td>QB2</td><td>Q2. 7</td><td>Q2. 6</td><td>Q2. 5</td><td>Q2. 4</td><td>Q2. 3</td><td>Q2. 2</td><td>Q2. 1</td><td>Q2. 0</td></tr>
<tr><td rowspan="4">QD3</td><td rowspan="2">QW3</td><td>QB3</td><td>Q3. 7</td><td>Q3. 6</td><td>Q3. 5</td><td>Q3. 4</td><td>Q3. 3</td><td>Q3. 2</td><td>Q3. 1</td><td>Q3. 0</td></tr>
<tr><td rowspan="4">QD4</td><td rowspan="2">QW4</td><td>QB4</td><td>Q4. 7</td><td>Q4. 6</td><td>Q4. 5</td><td>Q4. 4</td><td>Q4. 3</td><td>Q4. 2</td><td>Q4. 1</td><td>Q4. 0</td></tr>
<tr><td rowspan="4">QD5</td><td rowspan="2">QW5</td><td>QB5</td><td>Q5. 7</td><td>Q5. 6</td><td>Q5. 5</td><td>Q5. 4</td><td>Q5. 3</td><td>Q5. 2</td><td>Q5. 1</td><td>Q5. 0</td></tr>
<tr><td rowspan="4">QD6</td><td rowspan="2">QW6</td><td>QB6</td><td>Q6. 7</td><td>Q6. 6</td><td>Q6. 5</td><td>Q6. 4</td><td>Q6. 3</td><td>Q6. 2</td><td>Q6. 1</td><td>Q6. 0</td></tr>
<tr><td rowspan="4">QD7</td><td rowspan="2">QW7</td><td>QB7</td><td>Q7. 7</td><td>Q7. 6</td><td>Q7. 5</td><td>Q7. 4</td><td>Q7. 3</td><td>Q7. 2</td><td>Q7. 1</td><td>Q7. 0</td></tr>
<tr><td rowspan="4">QD8</td><td rowspan="2">QW8</td><td>QB8</td><td>Q8. 7</td><td>Q8. 6</td><td>Q8. 5</td><td>Q8. 4</td><td>Q8. 3</td><td>Q8. 2</td><td>Q8. 1</td><td>Q8. 0</td></tr>
<tr><td rowspan="4">QD9</td><td rowspan="2">QW9</td><td>QB9</td><td>Q9. 7</td><td>Q9. 6</td><td>Q9. 5</td><td>Q9. 4</td><td>Q9. 3</td><td>Q9. 2</td><td>Q9. 1</td><td>Q9. 0</td></tr>
<tr><td rowspan="4">QD10</td><td rowspan="2">QW10</td><td>QB10</td><td>Q10. 7</td><td>Q10. 6</td><td>Q10. 5</td><td>Q10. 4</td><td>Q10. 3</td><td>Q10. 2</td><td>Q10. 1</td><td>Q10. 0</td></tr>
<tr><td rowspan="4">QD11</td><td rowspan="2">QW11</td><td>QB11</td><td>Q11. 7</td><td>Q11. 6</td><td>Q11. 5</td><td>Q11. 4</td><td>Q11. 3</td><td>Q11. 2</td><td>Q11. 1</td><td>Q11. 0</td></tr>
<tr><td rowspan="4">QD12</td><td rowspan="2">QW12</td><td>QB12</td><td>Q12. 7</td><td>Q12. 6</td><td>Q12. 5</td><td>Q12. 4</td><td>Q12. 3</td><td>Q12. 2</td><td>Q12. 1</td><td>Q12. 0</td></tr>
<tr><td></td><td rowspan="2">QW13</td><td>QB13</td><td>Q13. 7</td><td>Q13. 6</td><td>Q13. 5</td><td>Q13. 4</td><td>Q13. 3</td><td>Q13. 2</td><td>Q13. 1</td><td>Q13. 0</td></tr>
<tr><td></td><td></td><td rowspan="2">QW14</td><td>QB14</td><td>Q14. 7</td><td>Q14. 6</td><td>Q14. 5</td><td>Q14. 4</td><td>Q14. 3</td><td>Q14. 2</td><td>Q14. 1</td><td>Q14. 0</td></tr>
<tr><td></td><td></td><td></td><td></td><td>QB15</td><td>Q15. 7</td><td>Q15. 6</td><td>Q15. 5</td><td>Q15. 4</td><td>Q15. 3</td><td>Q15. 2</td><td>Q15. 1</td><td>Q15. 0</td></tr>
</table>

以字节使用时，符号数值范围是：－128 ~ ＋127；
以字节使用时，非符号数值范围是：0 ~ 255；
以字使用时，符号数值范围是：－32768 ~ ＋32767；
以字使用时，非符号数值范围是：0 ~ 65535；
以双字使用时，符号数值范围是：－2147483648 ~ ＋2147483647；
以双字使用时，非符号数值范围是：0 ~ 4294966295。

2.2　M/S 中间继电器/状态继电器

问：在程序中看到有 M、SM 和 S 的符号，如何理解这些软元件?

问：在程序中看到有 M、SM 和 S 的符号，如何理解这些软元件?

答： M 是中间继电器，SM 是特殊继电器，S 是状态继电器。下面分别论述这些软元件。

中间继电器 M 的编号及属性见表 2-3，中间继电器一般用来存储程序中的中间状态或控制信息，中间继电器在编程中可以按位 M0.0、字节 MB10、字 MW10 或双字 MD10 来使用。

表 2-3　中间继电器 M 的编号及属性表

一般用途(默认)	停电保持用(默认)
M0.0 ~ M13.7	M14.0 ~ M31.7

PLC 内有的 M 其模型是由一个线圈和一对触点组成。其线圈可由 PLC 内的各种软元件的触点驱动。这类 M 有无数的常开触点（—┤├—）与常闭触点（—┤/├—），在 PLC 内可随意使用。但是 M 触点不能直接驱动外部负载，外部物理负载只能通过 Q 驱动。

一般用途（非停电保持用途）通过修改系统块中保留性范围参数设定可以改为停电保持用途。同样，停电保持用途的也可以通过修改系统块中保留性范围参数设定改为一般用途。系统块中保留性范围参数设定如图 2-1 所示。



图 2-1　保留性范围参数设定

PLC 在运行过程中如果突然停电，Q 及一般 M 都会断开，当重新上电运行时，除输入条件为接通的情况以外，都为断开状态。但在实际控制过程中，如果希望记忆停电前的状态，就需用停电保持型 M。

把图 2-2 所示程序的 M14.0 改为普通用途型 M0.0（默认状态是非停电保持用途），如图 2-3 所示，启动/停止动作过程基本相同，不同的是当 PLC 停电，再上电恢复到运行状态时，M0.0 无法保持停电前的接通状态。

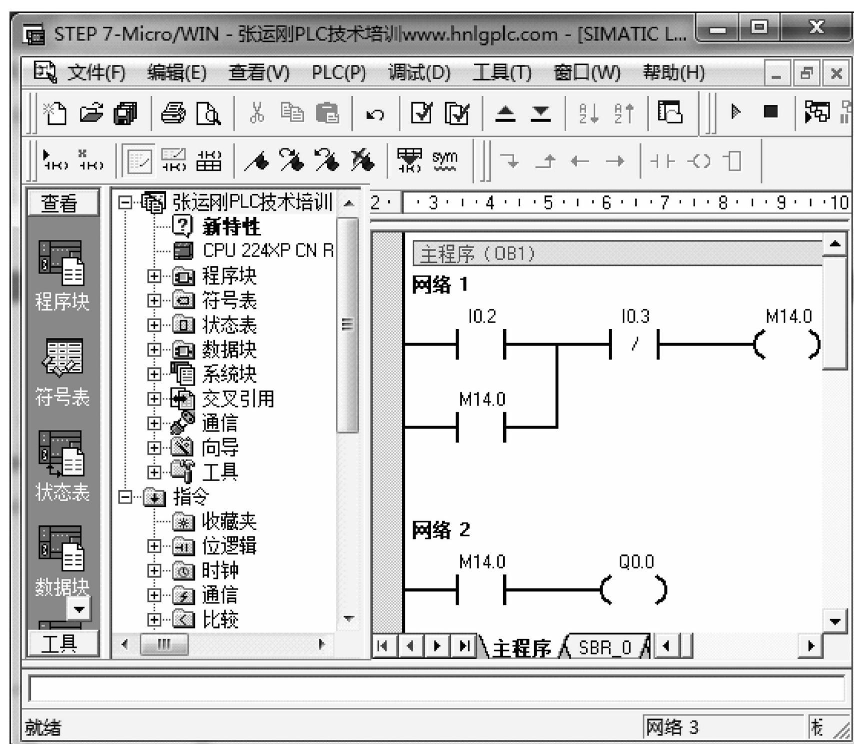


图 2-2　启动/停止控制程序

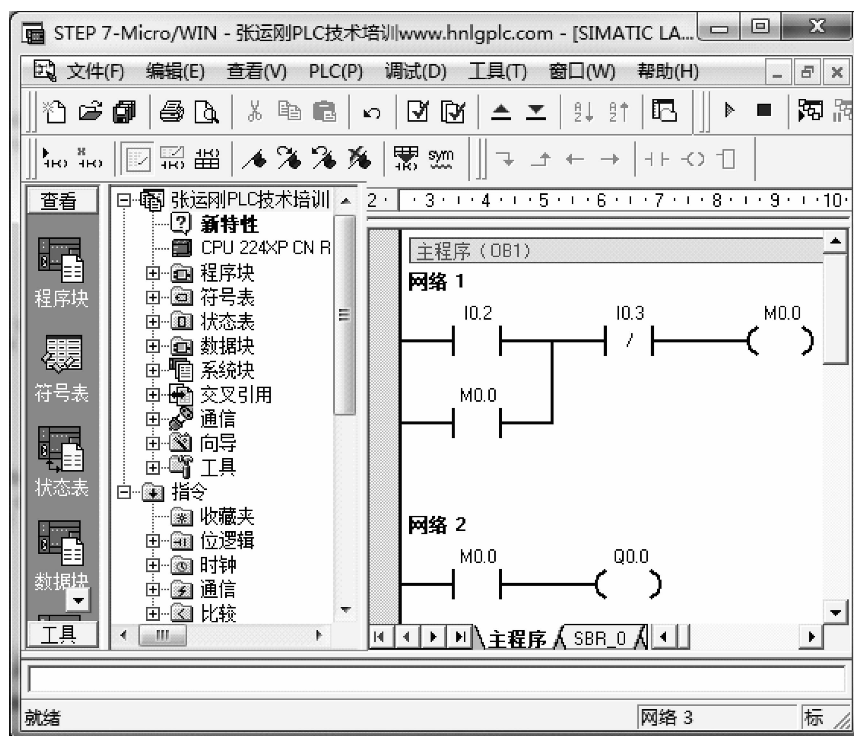


图 2-3　启动/停止控制程序

特殊中间继电器 SM 编号及属性见表 2-4，特殊中间继电器是 CPU 系统与用户程序之间互相交换信息的窗口。特殊中间继电器可以按位 SM0.0、字节 SMB10、字 SMW10 或双字 SMD10 来使用。

表 2-4　特殊中间继电器 SM 的编号及属性表

CPU 类型	所有范围	只读范围
CPU221	SM0.0 ~ SM179.7	SM0.0 ~ SM29.7
CPU222	SM0.0 ~ SM299.7	SM0.0 ~ SM29.7
CPU224	SM0.0 ~ SM549.7	SM0.0 ~ SM29.7
CPU226	SM0.0 ~ SM549.7	SM0.0 ~ SM29.7

SM0.1、SM0.5 和 SM0.0 是触点型特殊继电器。SM0.1 是开机脉冲，只在“STOP”转到“RUN”状态的第一个扫描周期是导通的；SM0.5 是 1s 脉冲；SM0.0 在“RUN”状态时总是接通的。

SMB30 是定义 PORT0 通信口工作模式的特殊中间继电器；SMB34 是定义定时中断间隔时间的特殊中间继电器；SMB47 是定义高速计数器 HSC1 工作模式的特殊中间继电器。更多的特殊继电器信息请查看帮助信息。

S 状态继电器的编号是 S0.0 ~ S31.7。状态 S 一般用来编写步进阶梯指令，表示该步的状态，配合 SCR 指令完成步进阶梯指令控制程序的逻辑分段。状态 S 在编程或调试时可以按位 S0.0、字节 SB10、字 SW10 或双字 SD10 来使用。

在不用 SCR 指令时，状态 S 与普通 M 功能一样，但是对工序步进控制编程时只能用状态 S 元件，与步进梯形图 SCR 指令结合使用（详细具体的应用请看步进阶梯指令章节内容）。

2.3　V/L 数据存储器/临时寄存器

问：S7-200 程序中的 V 和 L 符号代表什么符号？如何理解它们的属性？

问：S7-200 程序中的 V 和 L 符号代表什么符号？如何理解它们的属性？

答：V 是数据存储器，是全局变量；L 是临时数据寄存器，是局部变量。他们的属相描述如下。

S7-200 系列 PLC 数据存储器 V 的编号及属性见表 2-5。

表 2-5　S7-200 系列 PLC 数据存储器 V 的编号及属性

CPU221	CPU222	CPU224	CPU226	CPU226XM
VB0 ~ VB2047	VB0 ~ VB2047	VB0 ~ VB5119	VB0 ~ VB5119	VB0 ~ VB10239
注：默认全部是停电保持型，通过保留性范围参数设定可以改变保留性范围				

数据存储器 V 是存储执行程序过程中的中间结果或保存与工序或任务有关的数据的软元件。数据存储器 V 可以按位（V10.0）、字节（VB10）、字（VW10）或双字（VD10）来使用。按照字节（VB10）、字（VW10）或双字（VD10）来使用，如果认为是符号数，最高位为符号位（1 为负数，0 为正数）。字节（VB10）、字（VW10）或双字（VD10）来使用时其最高位是 V10.7。

数据存储器的功能分为一般用途型和停电保持型。默认状态全部都是停电保持型。默认

停电保持用途的也可以通过修改编程软件的系统块中保留性范围参数设定改为一般用途，比如若按照图 2-4 所示设置参数。VB0 ~ VB19 称为一般用途，VB20 以上是停电保持用途。

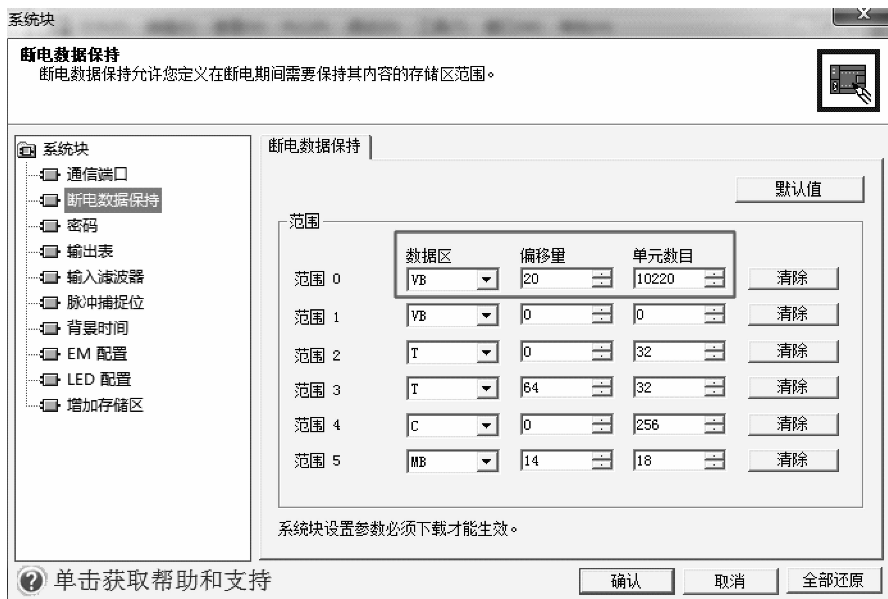


图 2-4　保留性范围参数设定

一般用途数据存储器如 VB10 一旦写入数据，其里面的数据不会自然变化，即使在 RUN →STOP→RUN 时也不会变化。但在停电时，所有数据会自动复位为“0”，应用程序如图 2-5 中的 VB10 所示。

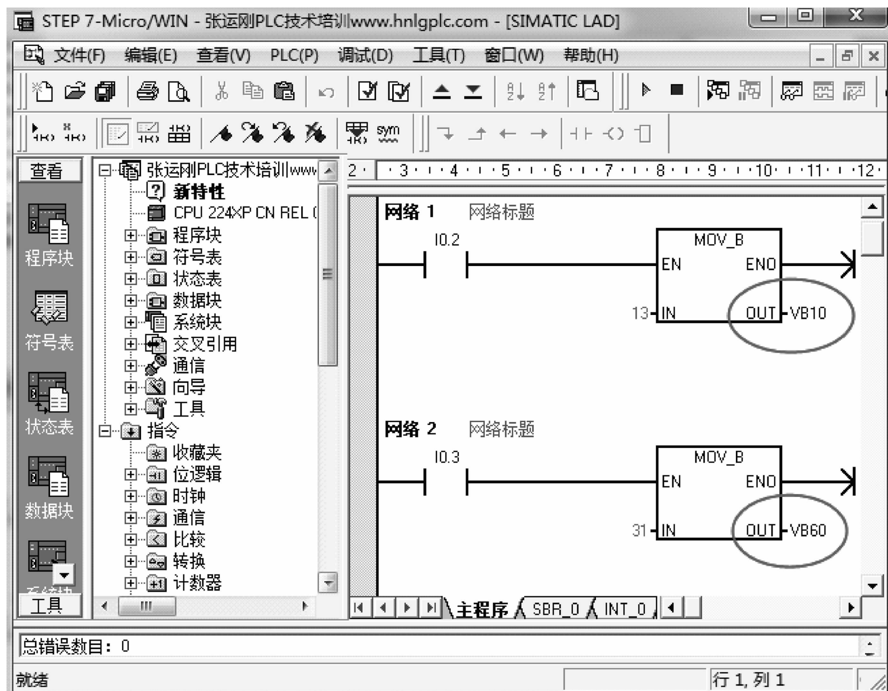


图 2-5　一般用途与保持用途的 V

停电保持型的数据存储器里面的数据如 VB60，若用户不对其内容进行更改，则里面的数据不会改变，无论是在 STOP 或停电状态，应用程序如图 2-2 中的 VB60 所示。

L 从保存数值属性角度看，其临时数据寄存器意思是只能临时保存数据，比如当 PLC 切换在 STOP 状态或者停电重新上电时，其数据会复位。当从应用领域角度看，是局部变量，意思是只能用于一个程序块，不能用于其他的程序块，当在其他程序块出现尽管地址和符号都相同但其代表是不同的变量。

临时数据寄存器 L 是存储执行程序过程中的中间临时结果。临时数据寄存器 L 可以按位（L10. 0）、字节（LB10）、字（LW10）或双字（LD10）来使用。

数据存储器 V 的状态可以在状态表监控，而 L 状态则不能。L 状态只能在程序界面，在“使用执行状态”模式的监控程序状态，可以监控到临时数据寄存器 L 变量的状态，如图 2-6 所示。

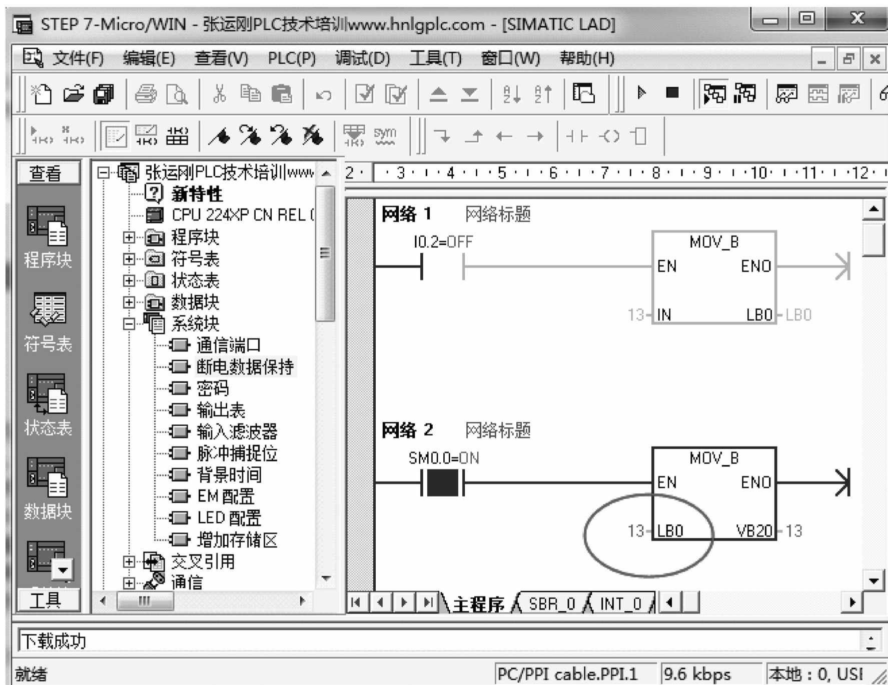


图 2-6　监控 L 状态

2. 4　常量和数制

问：S7-200 支持什么进制数据？支持中文字符类型吗？怎样理解这些数据和符号？

问：S7-200 支持什么进制数据？支持中文字符类型吗？怎样理解这些数据和符号？

答：S7-200 支持二进制、十进制和十六进制数，支持 ASCII 字符也支持中文字符。这些数制和字符详细描述如下。

二进制数据中，字节是 8 位、字是 16 位，双字是 32 位。十进制和十六进制数据见表 2-6。

表 2-6　十进制和十六进制数值范围

数据大小	不带符号的整数范围		带符号的整数范围	
	十进制数字	十六进制数字	十进制数字	十六进制数字
B(字节)	0～255	0～FF	0～255	0～FF
W(字)	0～65535	0～FFFF	－32768～＋32767	8000～7FFF
D(双字)	0～4294967295	0～FFFF FFFF	－2147483648～＋2147483647	80000000～7FFF FFFF
	十进制小数(正数)		十进制小数(负数)	
D(双字)	+1.175495E－38～＋3.402823E＋38		－1.175495E－38～－3.402823E＋38	

数据格式标识符。在许多指令中使用字节、字或双字大小的数据，格式标识符显示这些常量是二进制数字、十进制数字、十六进制数字或是 ASCII 字符。

程序中默认常量为十进制数字，除非使用格式标识符：

2#　　二进制数字；

16#　　十六进制数字。

ASCII 常量。使用单引号字符起始与结束 ASCII 字符。对在参数列表中指定常量的绝大多数指令有效，作为数据字节存储。

ASCII 常量字符串。使用双引号字符起始与结束 ASCII 字符。对在参数列表中指定常量字符串的绝大多数指令有效，作为长度字节后接数据字节存储。

二进制数据举例见表 2-7。

表 2-7　二进制数据举例

	二进制数据	数字基数	分隔符	常量数值
例 1	2#1101	2	#	1101
例 2	2#1101_1111	2	#	1101_1111

十六进制数据见表 2-8。

表 2-8　十六进制数据举例

	十六进制数据	数字基数	分隔符	常量数值
例	16#3FB2	16	#	3FB2
例	16#3_F_B_2	注：可使用下划线，以便增强可读性		

模拟量数据极限范围为－32000～＋32000。在实际用于中常用的模拟量有 13 位、14 位和 16 位，其分辨率是不同的，见表 2-9 所示。

表 2-9　不同位的模拟量分辨率

位数	分辨率		模拟值	
	十进制	十六进制	高字节	低字节
13	8	16#8	符号 xxxxxxx	xxxxx000
14	4	16#4	符号 xxxxxxx	xxxxxx00
16	1	16#1	符号 xxxxxxx	xxxxxxxx

ASCII 常量字符范围：

ASCII 常量字符的有效范围为 ASCII 32 ~ ASCII 255，但不包括 DEL（删除）字符、单引号字符、双引号字符。在此范围之外的 ASCII 字符必须使用特殊$字符格式。

常量字符特例见表 2-10 所示。

表 2-10　常量字符

字符	字节 1	字节 2	字节 3	字节 4	字节 5
	字符 1	字符 2	字符 3	字符 4	字符 5
	VB1	VB2	VB3	VB4	VB5
A	A				
AB	A	B			
ABC	A	B	C		
ABCD	A	B	C	D	
ABCDE	A	B	C	D	E

ASCⅡ常数字符串：

字符串是一系列字符组合，每个字符需要一个字节存储。字符串的第一个字节是定义字符串的长度，或者说称字符数。字符串的长度可为 0 ~ 254 个字符。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符来表达起始和结束，如“字符串常数”。

ASCⅡ常数字符串数据类型的格式：

表 2-11 表示了字符串数据类型的格式。字符串最长为 255 个字节（254 个字符加长度字节）

表 2-11　字符串数据类型的格式

字节 0	字节 1	字节 2	字节 3	字节 4	…	字节 254
字符长度	字符 1	字符 2	字符 3	字符 4	…	字符 254

不同长度字符串表达见表 2-12 所示。

表 2-12　不同长度字符串表达法

字符串	字节 0	字节 1	字节 2	字节 3	字节 4	字节 5
	长度	字符 1	字符 2	字符 3	字符 4	字符 5
	VB0	VB1	VB2	VB3	VB4	VB5
“A”	1	A				
“AB”	2	A	B			
“ABC”	3	A	B	C		
“ABCD”	4	A	B	C	D	
“ABCDE”	5	A	B	C	D	E

第3章　基 本 指 令

3.1　一个开关驱动一个输出

问：一个开关控制一个输出的程序怎样编写？ 如何调试和监控程序？

问1：一个开关控制一个输出的程序怎样编写？

答：一个开关控制一个输出的程序，使用继电器驱动逻辑编写程序，如图3-1所示，I0.2驱动Q0.3。



图3-1　一个开关控制一个输出的程序

图3-1是梯形图指令，其形式非常像电工里的梯形图。当PLC在运行状态（RUN）I0.2接通时，Q0.3也接通了；当I0.2断开，Q0.3也断开。控制动作就像房间里的开关控制房灯一样，I0.2就是开关，Q0.3就是房灯。

问2：如何调试和监控程序？

答：调试和监控前，首先把程序下载到CPU，然后让CPU工作在RUN模式。

调试程序的时候，进入监控状态，在I0.2接通时梯形图的状态如图3-2所示，图中“┤▒├”代表输入接通，“-(▒)”代表输出接通。I0.2接通驱动Q0.3，意思是I0.2接通

Q0.3 也接通。

在调试的时候，进入监控状态，在 I0.2 断开时梯形图的状态如图 3-3 所示，图中“┤ ├”代表输入没有接通，“-( )”代表输出没有接通。I0.2 断开不驱动 Q0.3，意思是 I0.2 断开 Q0.3 也断开。

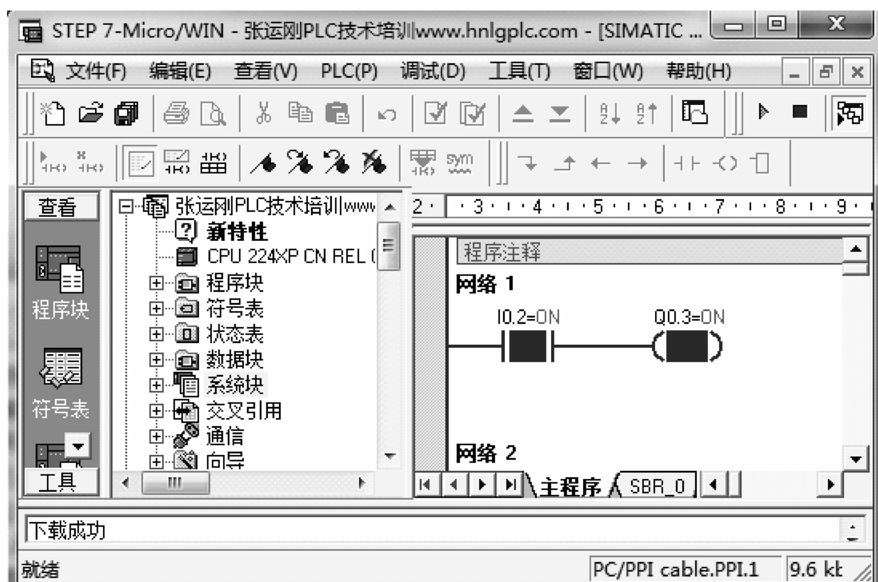


图 3-2　开关接通时的程序状态

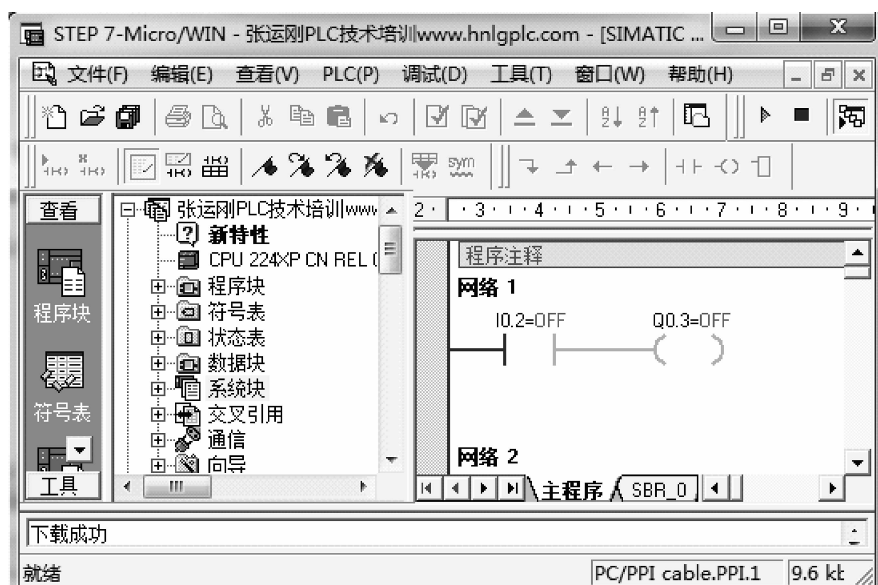


图 3-3　开关断开时的程序状态

3.2　一个开关置位/复位输出

> 问：一个开关控制一个输出，使用“SET”和“RST”指令的程序怎样编写？
> 如何调试和监控程序？

问1：一个开关控制一个输出，使用“SET”和“RST”指令的程序怎样编写？

答：一个开关控制输出的程序，使用“SET”置位和“RST”复位逻辑编写程序如图3-4所示，I0.2控制Q0.3。

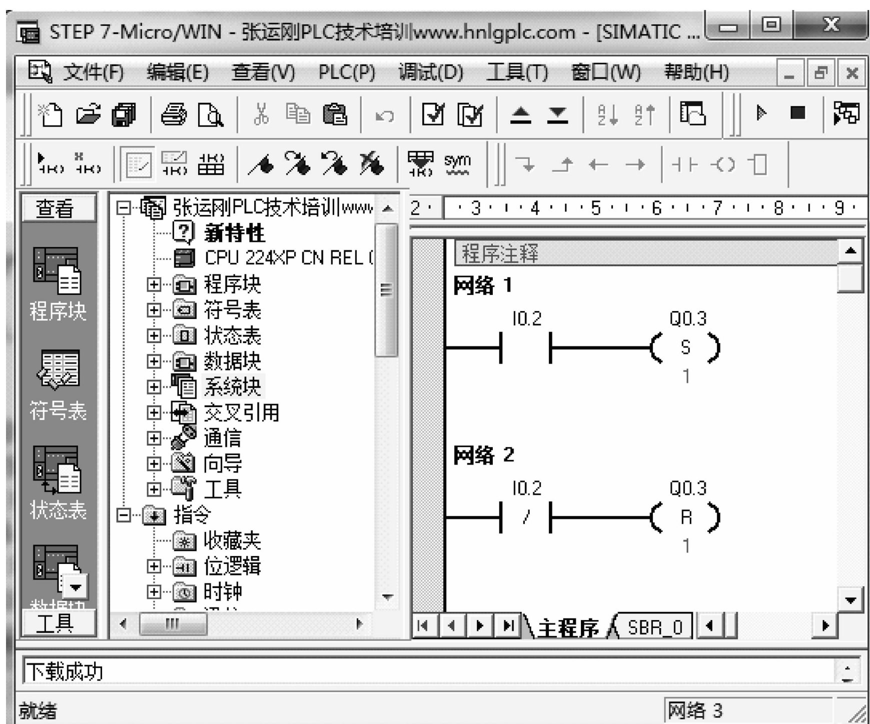


图3-4　一个开关控制输出的程序

问2：如何调试和监控程序？

答：调试前首先把程序下载到CPU，然后让CPU工作在RUN模式。

调试程序的时候，进入监控状态，在I0.2接通时梯形图的状态如图3-5所示，图中“┤█├”代表输入接通，“-(S)”代表输出接通。意思是I0.2接通置位Q0.3=1，也就是I0.2接通使Q0.3也接通。

在调试的时候，进入监控状态，在I0.2断开时梯形图的状态如图3-6所示，图中“┤/├”代表输入没有接通，“-(R)”代表输出没有接通。意思是I0.2断开复位Q0.3=0，也就是I0.2断开使Q0.3也断开。

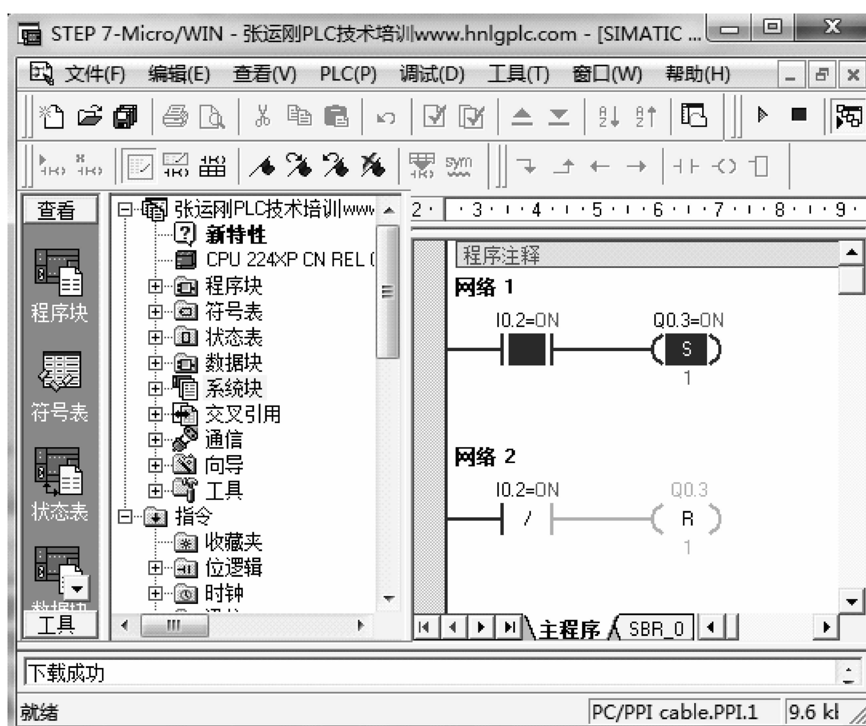


图 3-5　开关接通时的程序状态

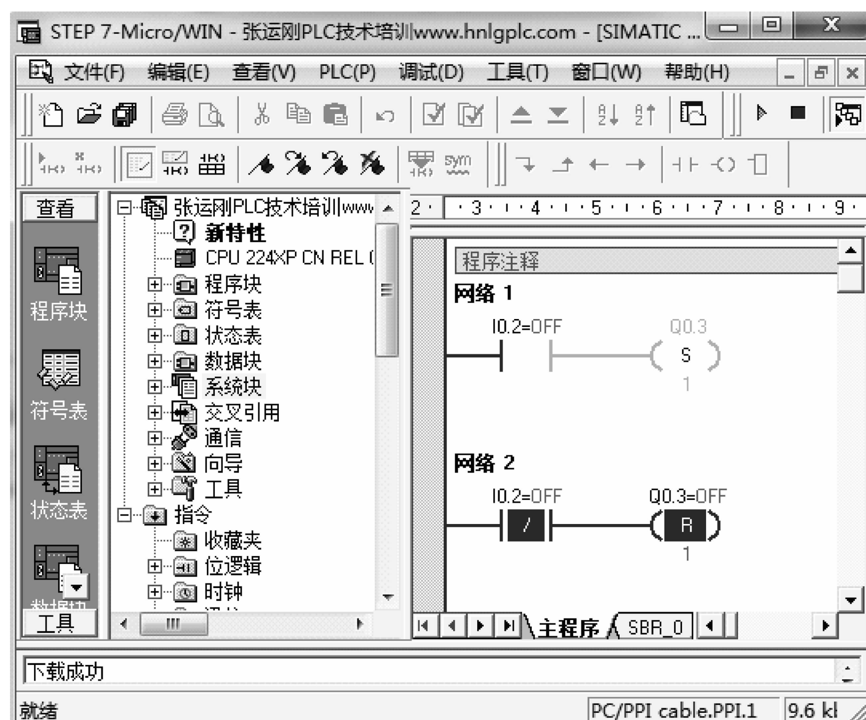


图 3-6　开关断开时的程序状态

3.3　一个开关脉冲沿置位/复位输出

问：一个开关使用脉冲沿控制一个输出，程序怎样编写？如何调试和监控？

问：一个开关使用脉冲沿控制一个输出，程序怎样编写？如何调试和监控？

答：一个开关使用脉冲沿控制输出的程序，使用“SET”置位和“RST”复位逻辑编写程序如图 3-7 所示，I0.2 上升沿置位 Q0.3，I0.2 下降沿复位 Q0.3。

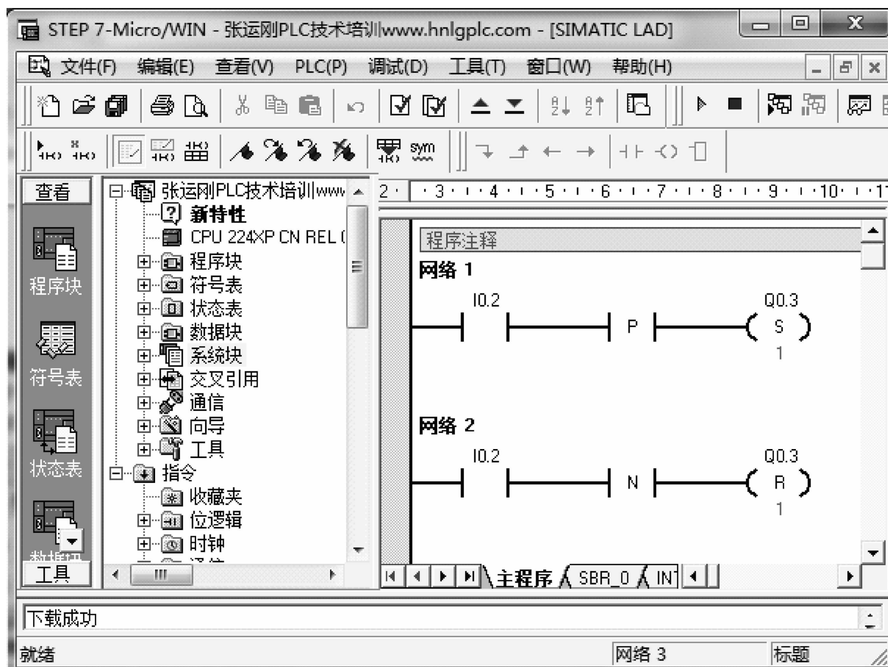


图 3-7　一个开关使用脉冲沿控制输出的程序

调试前首先把程序下载到 CPU，然后让 CPU 工作在 RUN 模式。

调试程序的时候，进入监控状态，在 I0.2 接通时梯形图的状态如图 3-8 所示，图中“⊣■⊢”代表输入接通，“-(S)”代表输出接通。I0.2 上升沿置位 Q0.3，意思是 I0.2 接通时使 Q0.3 = 1，也就是 I0.2 接通时使 Q0.3 也接通。

在调试的时候，进入监控状态，在 I0.2 断开时梯形图的状态如图 3-9 所示，图中“⊣ ⊢”代表输入没有接通，“-(R)”代表输出没有接通。I0.2 下降沿复位位 Q0.3，意思是 I0.2 断开时使 Q0.3 = 0，也就是 I0.2 断开时使 Q0.3 也断开。

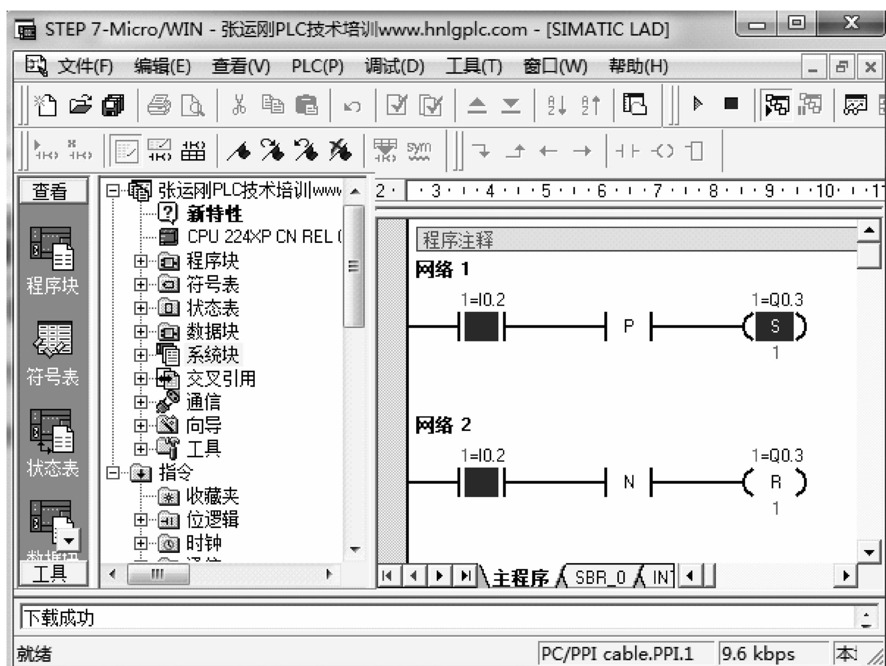


图 3-8　开关接通时的程序状态

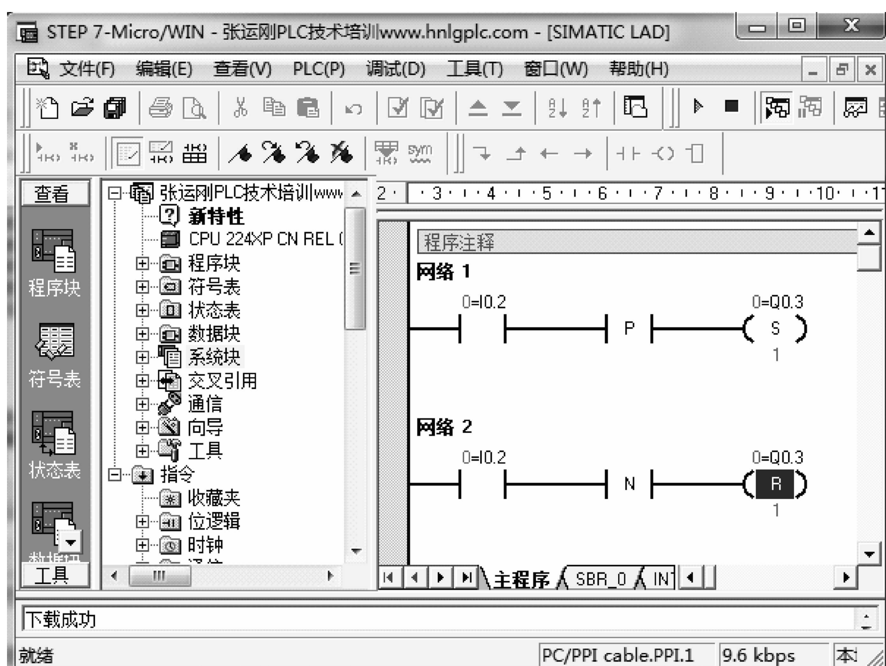


图 3-9　开关断开时的程序状态

3.4 启动按钮/停止按钮/保持/驱动输出

问：使用两个按钮控制一个输出的程序怎样编写？
如何调试和监控程序？

问1：使用两个按钮控制一个输出的程序怎样编写？

答：两个按钮，一个启动按钮和一个停止按钮，按照启动/保持/停止的逻辑编写程序如图3-10所示，I0.2按钮启动Q0.3，I0.3按钮停止Q0.3。

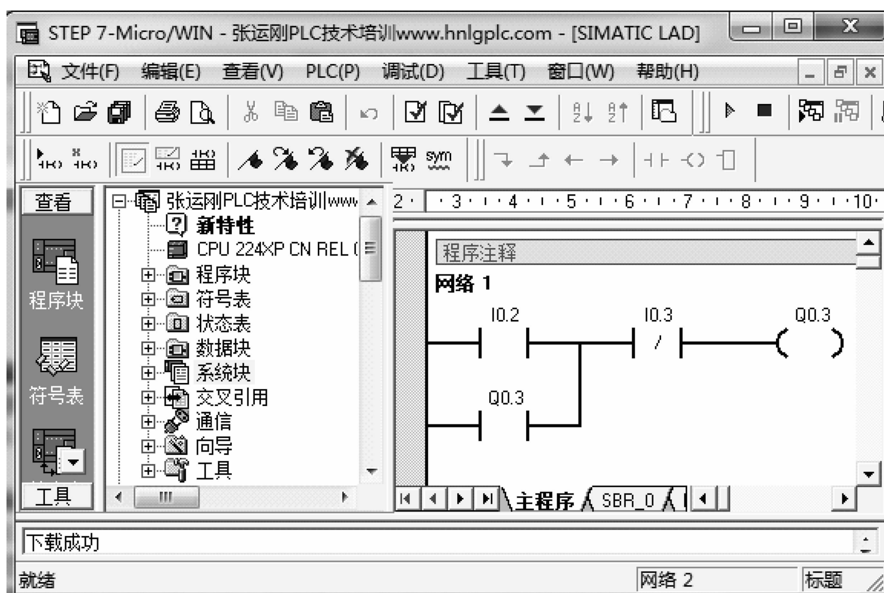


图3-10 启保停逻辑程序

问2：如何调试和监控程序？

答：调试前首先把程序下载到CPU，然后让CPU工作在RUN模式。

调试程序的时候，进入监控状态，按动I0.2按钮时梯形图的状态如图3-11所示，Q0.3是在接通状态。其控制原理是，在按动I0.2前Q0.3是断开状态，而I0.3的常闭触点一直都在接通状态。当I0.2接通时，I0.2常开触点与I0.3常闭触点是串联形式如图3-12所示，串联形式是"与"逻辑，当"与"的两个条件都满足，其输出就为1，这时候Q0.3接通了。当I0.2断开时，由于Q0.3常开触点现在闭合了，Q0.3触点与I0.2是并联形式如图3-13所示，并联形式是"或"逻辑，当"或"的两个条件其中有一个以上满足，其输出都是接通的，这时Q0.3常闭触点接通维持了Q0.3线圈一直在驱动状态，形成了自保持的逻辑关系。

在调试的时候，进入监控状态，按动I0.3程序状态如图3-14所示，图中Q0.3是在断开状态。其控制原理简单，当按动I0.3时其常闭触点断开了，没有驱动Q0.3线圈，所以Q0.3断开。

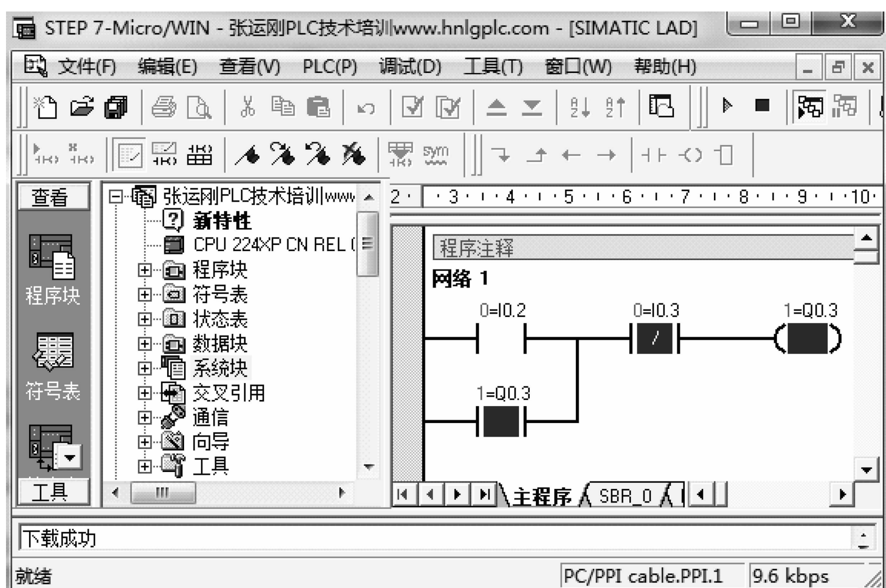


图 3-11　启动时的程序状态

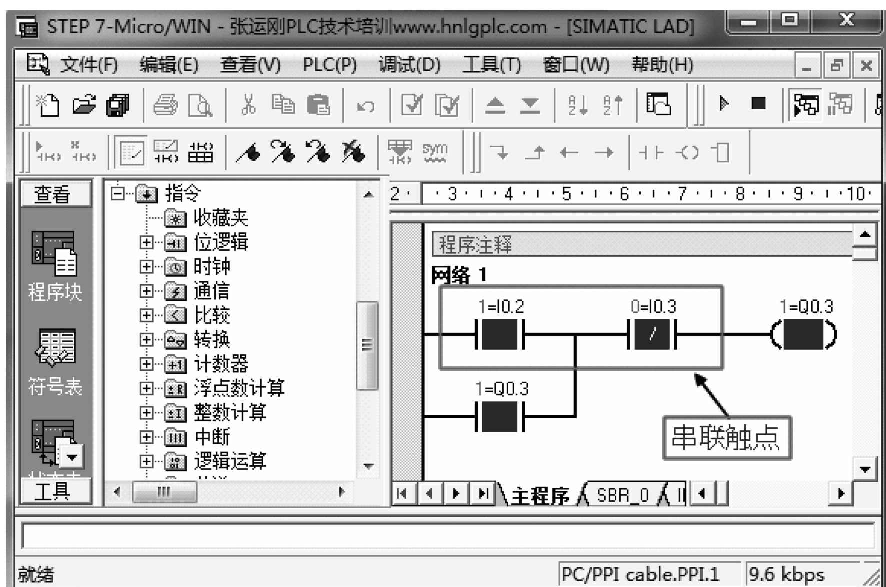


图 3-12　串联触点逻辑

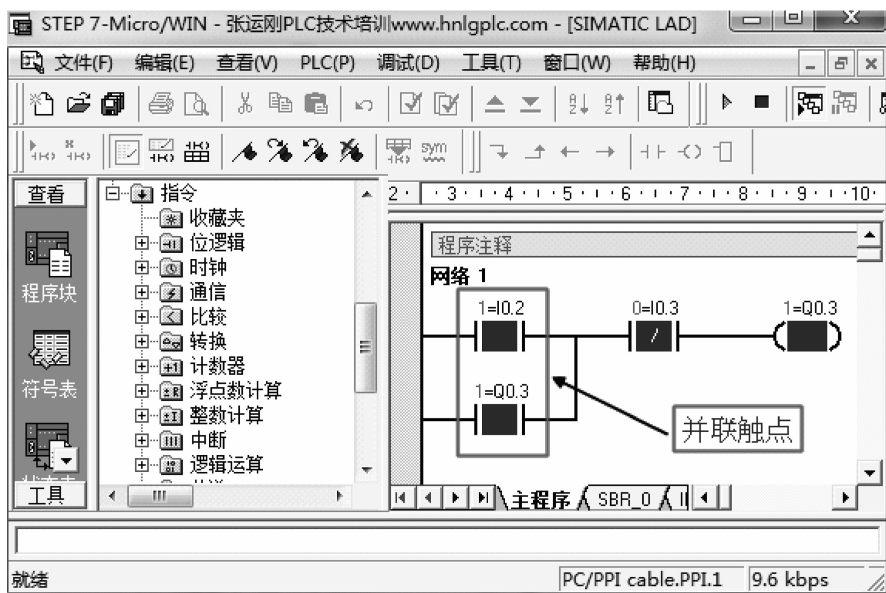


图 3-13 并联触点逻辑

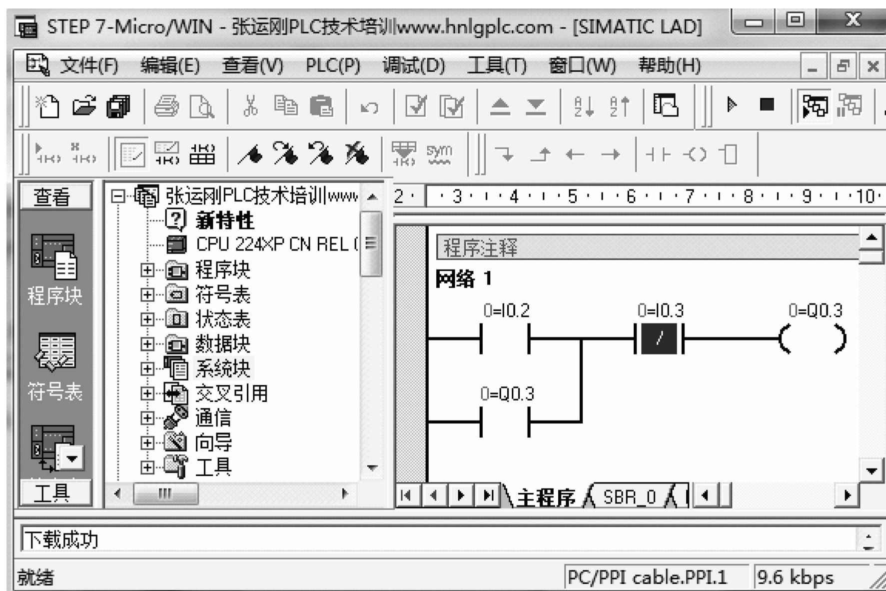


图 3-14 停止时的程序状态

3.5　启动按钮/停止按钮/置位/复位输出

> 问：两个按钮控制一个输出，使用“SET”和“RST”指令 RST 优先的程序怎样编写？如何调试和监控程序？
> 两个按钮控制一个输出，使用“SET”和“RST”指令 SET 优先的程序怎样编写？如何调试和监控程序？

问 1：两个按钮控制一个输出，使用“SET”和“RST”指令 RST 优先的程序怎样编写？

答：两个按钮控制一个输出的程序，使用“SET”置位和“RST”复位逻辑编写的程序如图 3-15 所示，这是复位优先的控制逻辑，其输入与输出的逻辑关系见表 3-1。

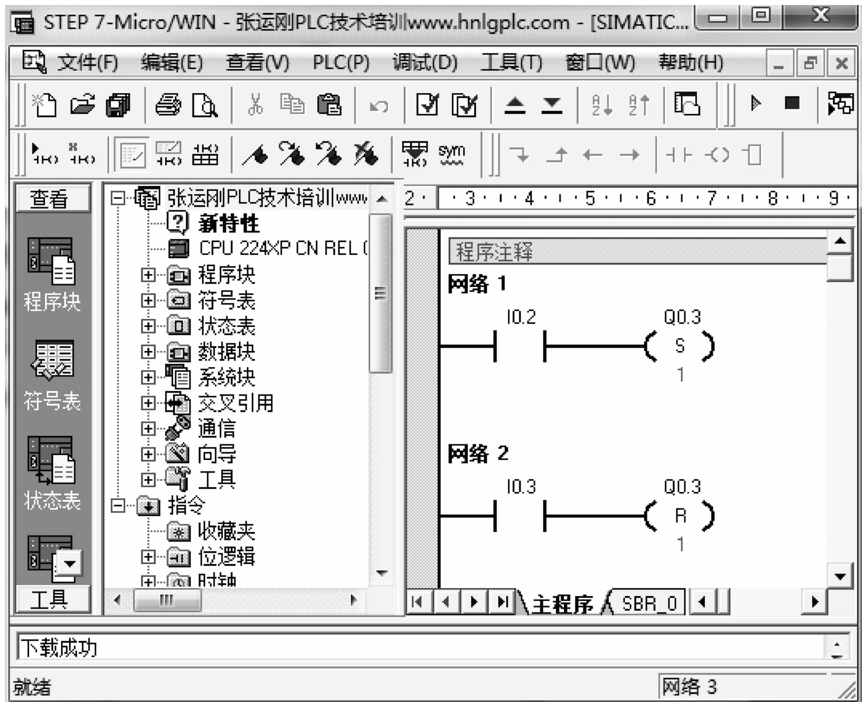


图 3-15　两个按钮控制输出的程序

表 3-1　复位优先输入与输出逻辑关系

I0.3(停止)	I0.2(启动)	Q0.3	I0.3(停止)	I0.2(启动)	Q0.3
0	0	保持	1	0	0
0	1	1	1	1	0

问 2：如何调试和监控程序？

答：调试前首先把程序下载到 CPU，然后让 CPU 工作在 RUN 模式。

启动。调试程序的时候，进入监控状态，在 I0.2 接通时梯形图的状态如图 3-16 所示，图中“┤■├”代表输入接通，“-(S)”代表输出接通。意思是 I0.2 接通置位 Q0.3 = 1，也就是 I0.2 接通使 Q0.3 也接通。

停止。在调试的时候，进入监控状态，在 I0.3 接通时梯形图的状态如图 3-17 所示，图中“┤■├”代表输入接通，“-(R)”代表复位输出了。意思是 I0.3 接通复位 Q0.3 = 0，也就是 I0.3 接通使 Q0.3 断开。

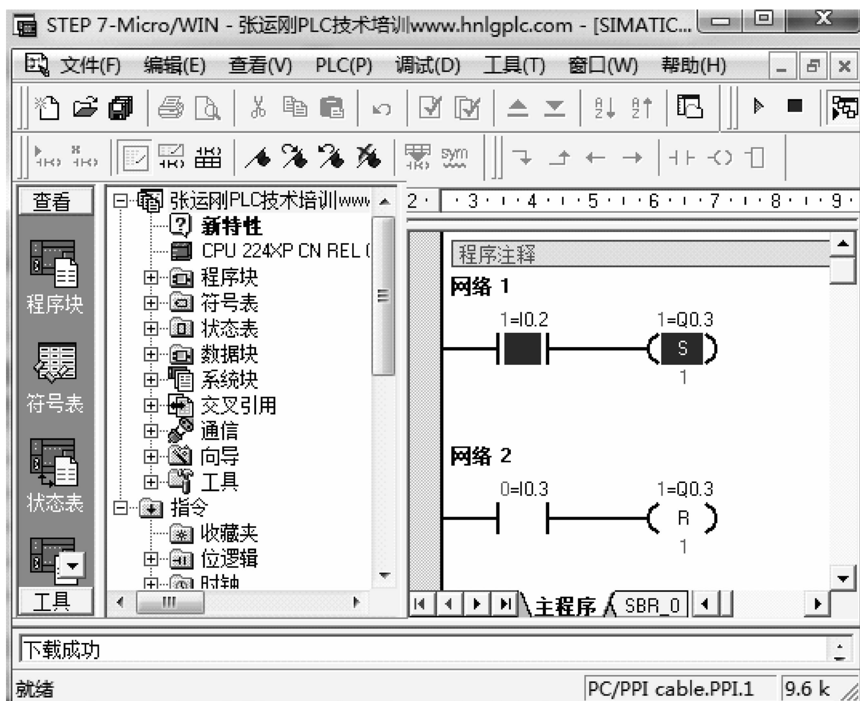


图 3-16　启动的程序状态

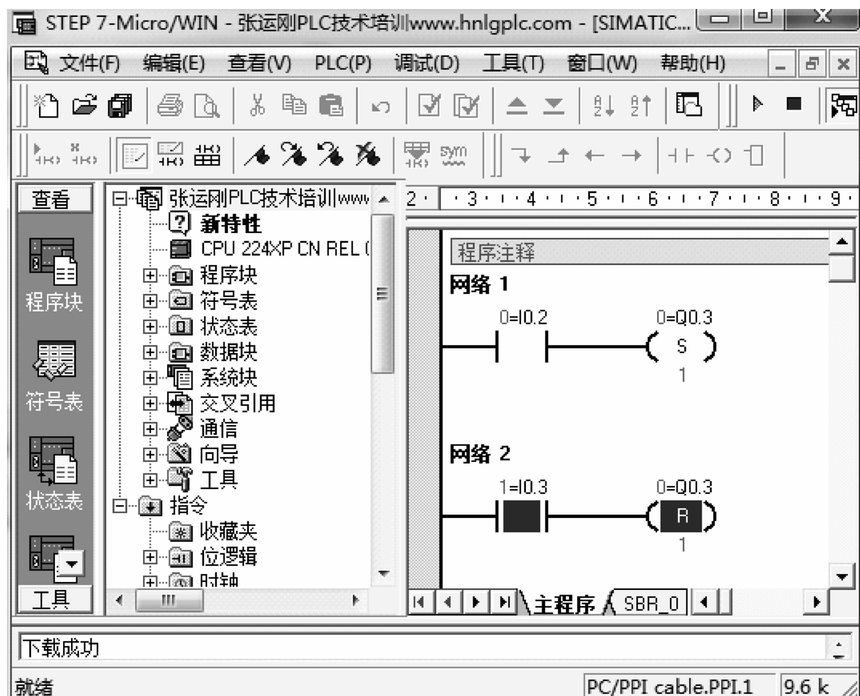


图 3-17　停止的程序状态

复位优先。当启动 I0.2 和停止 I0.3 两个按钮同时接通时，可以看到输出 Q0.3 是在断开状态，从这个逻辑上说是复位优先控制逻辑。

问 3：两个按钮控制一个输出，使用“SET”和“RST”指令 SET 优先的程序怎样编写？

两个按钮控制一个输出的程序，使用“SET”置位和“RST”复位逻辑编写的程序如图 3-18 所示，这是置位优先的控制逻辑，其输入与输出的逻辑关系见表 3-2。

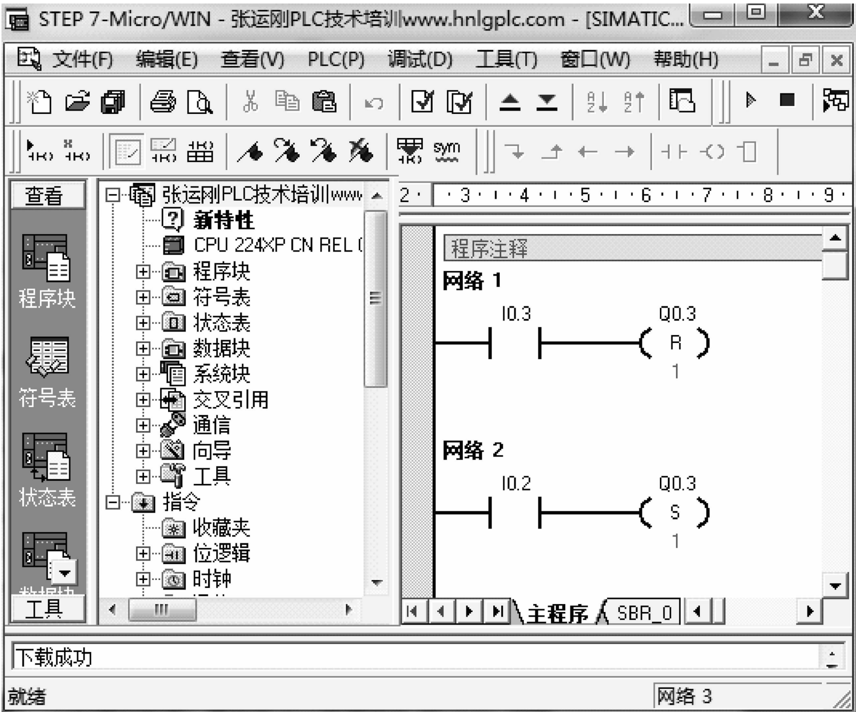


图 3-18　两个按钮控制输出的程序

表 3-2　置位优先输入与输出逻辑关系

I0.3(停止)	I0.2(启动)	Q0.3	I0.3(停止)	I0.2(启动)	Q0.3
0	0	保持	1	0	0
0	1	1	1	1	1

问 4：如何调试和监控程序？

答：调试前首先把程序下载到 CPU，然后让 CPU 工作在 RUN 模式。

启动。调试程序的时候，进入监控状态，在 I0.2 接通时梯形图的状态如图 3-19 所示，图中“┤■├”代表输入接通，“-(S)”代表输出接通。意思是 I0.2 接通置位 Q0.3 =1，也就是 I0.2 接通使 Q0.3 也接通。

停止。在调试的时候，进入监控状态，在 I0.3 接通时梯形图的状态如图 3-20 所示，图中“┤■├”代表输入接通，“-(R)”代表复位输出了。意思是 I0.3 接通复位 Q0.3 =0，也就是 I0.3 接通使 Q0.3 断开。

置位优先。当启动 I0.2 和停止 I0.3 两个按钮同时接通时，可以看到输出 Q0.3 是在接通状态如图 3-21 所示，从这个逻辑上说是置位优先控制逻辑。

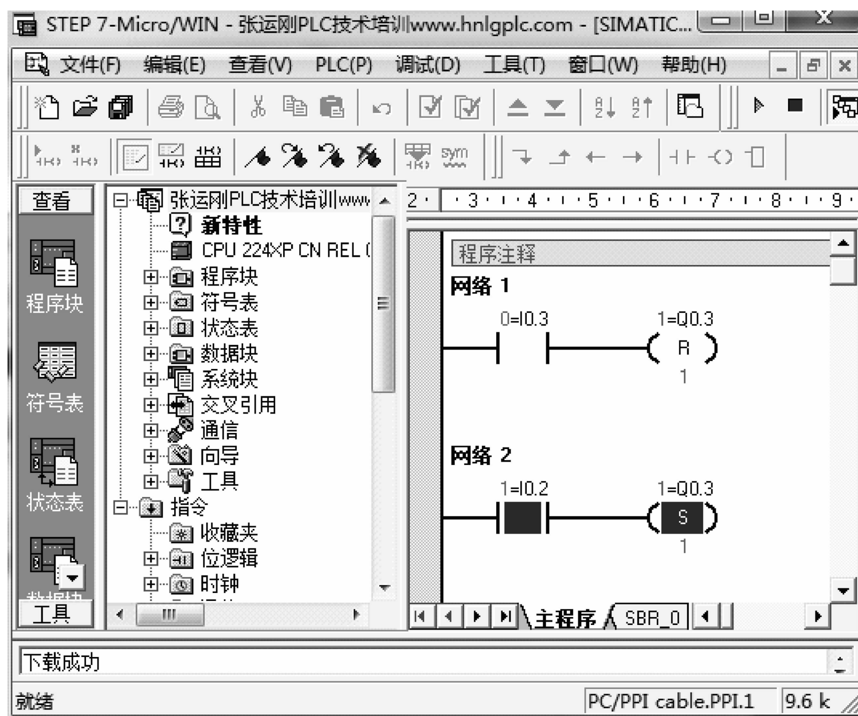


图 3-19 启动的程序状态

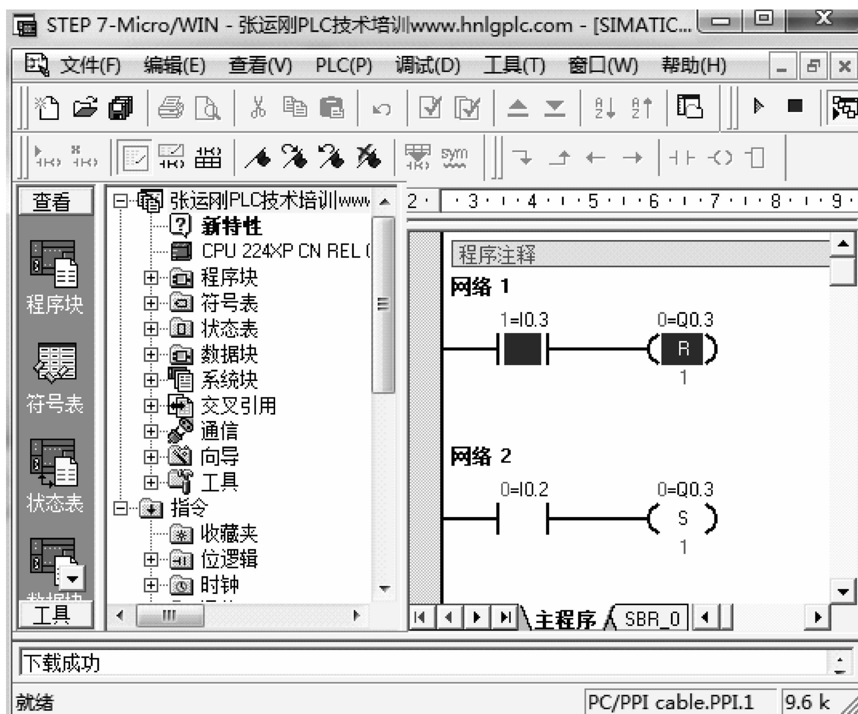


图 3-20 停止的程序状态

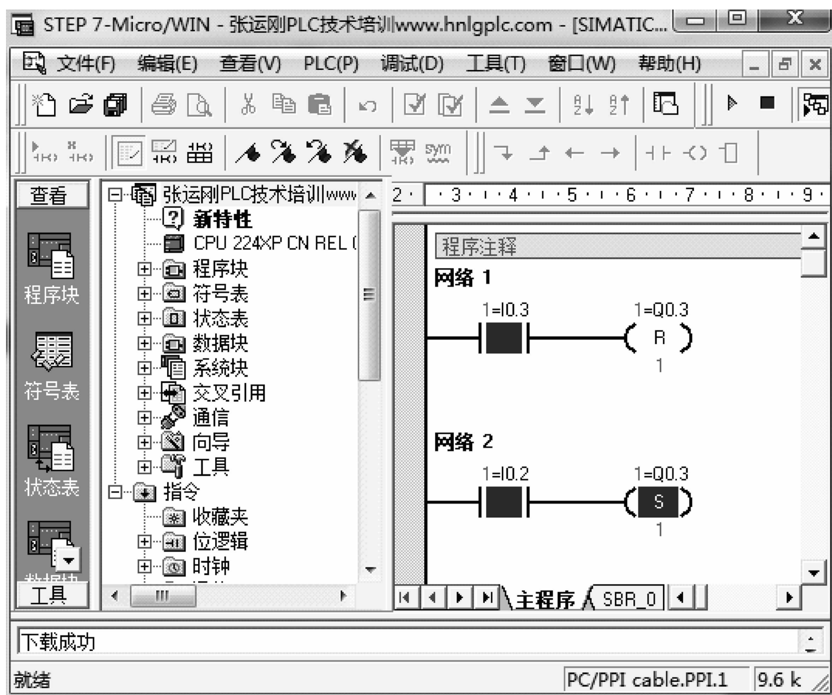


图 3-21　置位优先逻辑状态

3.6　启动按钮/停止按钮脉冲沿/置位/复位输出

问：两个按钮的脉冲沿控制一个输出，使用“SET”和“RST”指令 RST 优先的程序怎样编写？ 如何调试和监控程序？ 两个按钮的脉冲沿控制一个输出，使用“SET”和“RST”指令 SET 优先的程序怎样编写？ 如何调试和监控程序？

问 1：两个按钮的脉冲沿控制一个输出，使用“SET”和“RST”指令 RST 优先的程序怎样编写？

答：两个按钮的脉冲沿控制一个输出的程序，使用“SET”置位和“RST”复位逻辑编写的程序如图 3-22 所示，这是复位优先的控制逻辑，其输入与输出的逻辑关系见表 3-3。

问 2：如何调试和监控程序？

答：调试前首先把程序下载到 CPU，然后让 CPU 工作在 RUN 模式。

启动。调试程序的时候，进入监控状态，在 I0. 2 接通后梯形图的状态如图 3-23 所示，“-(S)”代表输出接通。意思是 I0. 2 接通上升沿置位 Q0. 3 = 1，也就是 I0. 2 接通瞬间使 Q0. 3 也接通。

停止。在调试的时候，进入监控状态，在 I0. 3 接通后梯形图的状态如图 3-24 所示，“-(R)”代表复位输出了。意思是 I0. 3 接通上升沿复位 Q0. 3 = 0，也就是 I0. 3 接通瞬间使 Q0. 3 断开。



图 3-22　两个按钮脉冲沿控制输出的程序

表 3-3　复位优先输入与输出逻辑关系

I0.3(停止)	I0.2(启动)	Q0.3	I0.3(停止)	I0.2(启动)	Q0.3
0	0	保持	后 1	先 1	0
0	1	1	先 1	后 1	1
1	0	0	同时为 1		0

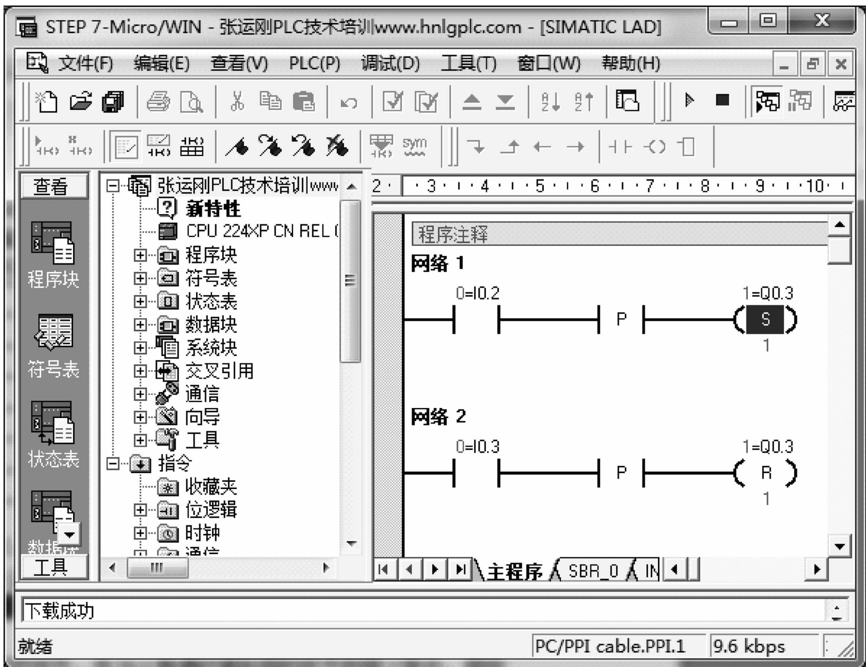


图 3-23　启动的程序状态

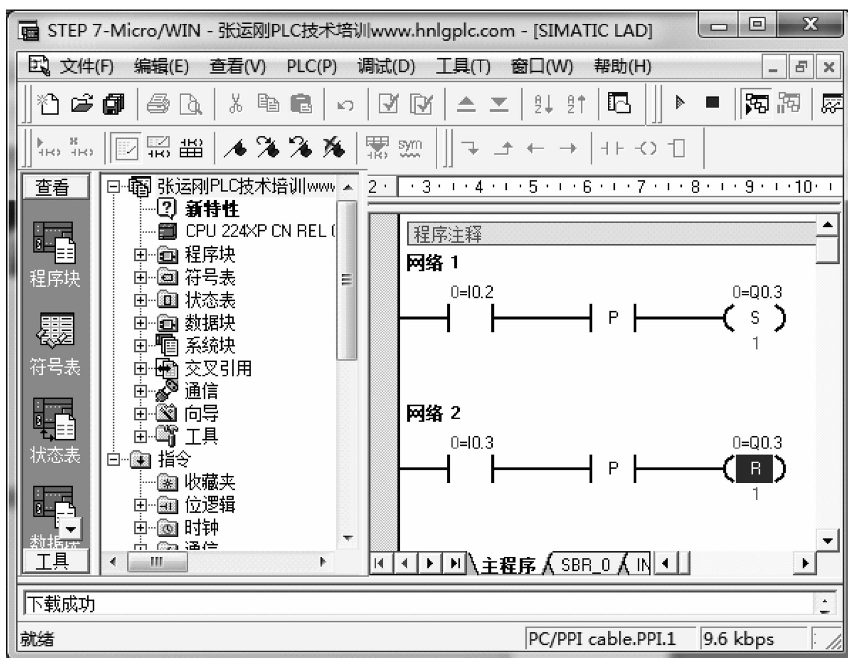


图 3-24　停止的程序状态

复位优先。当启动 I0.2 和停止 I0.3 两个按钮在同一个扫描周期接通时，可以看到输出 Q0.3 是在断开状态，从这个逻辑上说是复位优先控制逻辑。

问 3：两个按钮的脉冲沿控制一个输出，使用“SET”和“RST”指令 SET 优先的程序怎样编写？

两个按钮脉冲沿控制一个输出的程序，使用“SET”置位和“RST”复位逻辑编写的程序如图 3-25 所示，这是置位优先的控制逻辑，其输入与输出的逻辑关系见表 3-4。



图 3-25　两个按钮控制输出的程序

表 3-4　置位优先输入与输出逻辑关系

I0.3(停止)	I0.2(启动)	Q0.3	I0.3(停止)	I0.2(启动)	Q0.3
0	0	保持	后 1	先 1	0
0	1	1	先 1	后 1	1
1	0	0	同时为 1		1

问 4：如何调试和监控程序？

答：调试前首先把程序下载到 CPU，然后让 CPU 工作在 RUN 模式。

启动。调试程序的时候，进入监控状态，在 I0.2 接通后梯形图的状态如图 3-26 所示，“-(S)”代表输出接通。意思是 I0.2 接通上升沿置位 Q0.3 = 1，也就是 I0.2 接通瞬间使 Q0.3 也接通。



图 3-26　启动的程序状态

停止。在调试的时候，进入监控状态，在 I0.3 接通后梯形图的状态如图 3-27 所示，“-(R)”代表复位输出了。意思是 I0.3 接通上升沿复位 Q0.3 = 0，也就是 I0.3 接通瞬间使 Q0.3 断开。

置位优先。当启动 I0.2 和停止 I0.3 两个按钮在同一个扫描周期接通时，可以看到输出 Q0.3 是在接通状态，从这个逻辑上说是置位优先控制逻辑。



图 3-27　停止的程序状态

3.7　一个按钮控制一个输出

问：控制逻辑如图 3-28 所示，一个按钮控制一个输出的程序怎样编写？

问：控制逻辑如图 3-28 所示，一个按钮控制一个输出的程序怎样编写？

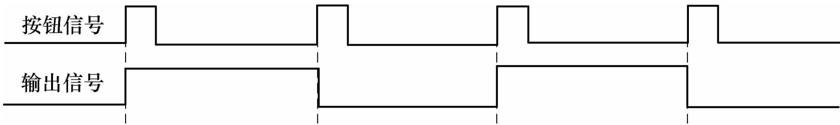


图 3-28　控制工艺逻辑时序图

答：一个按钮控制一个输出的控制逻辑方法很多，下面给出其中的 3 种供大家参考，如图 3-29 ~ 图 3-31 所示。

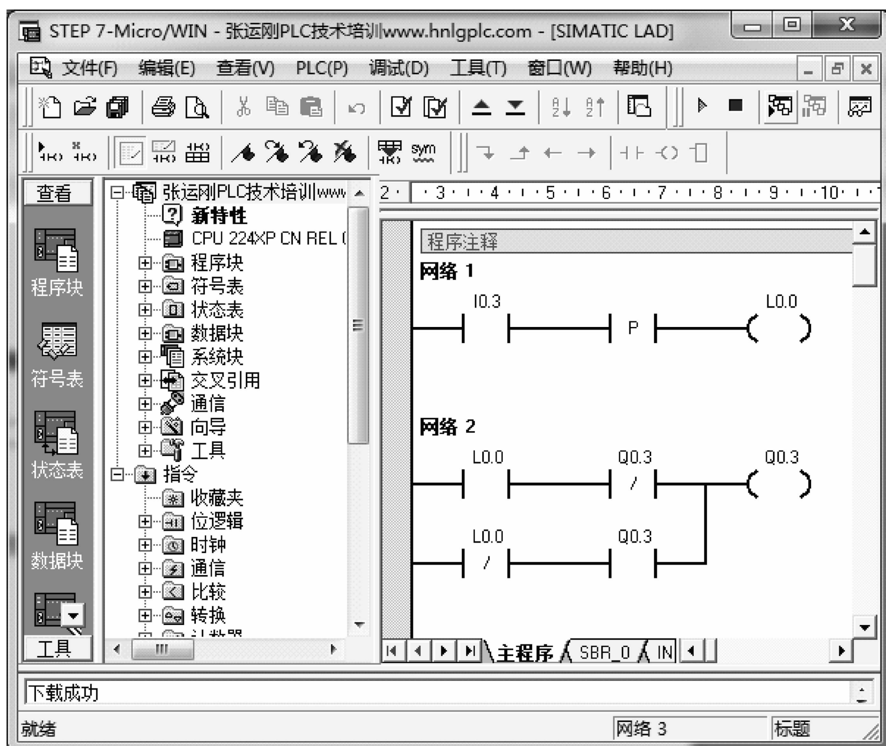


图 3-29 经验逻辑法

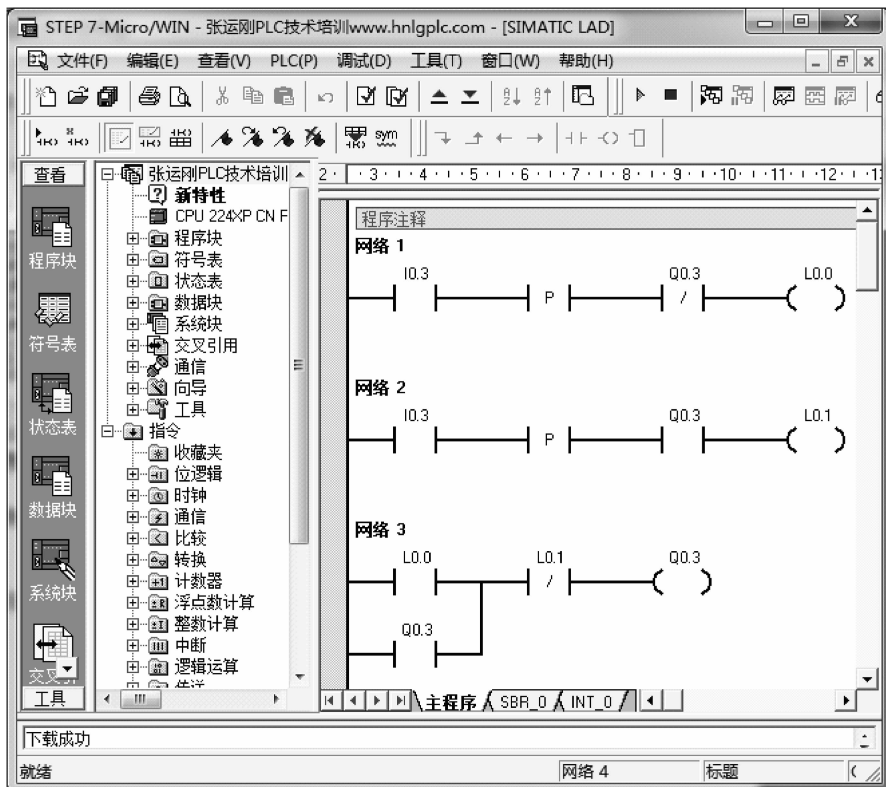


图 3-30 控制原理驱动法



图 3-31　PLC 扫描顺序法

3.8　步进阶梯指令

> 问：什么时候使用步进阶梯指令编程有优势？
> 如何理解步的状态？
> 步进阶梯指令怎样编程？

问 1：什么时候使用步进阶梯指令编程有优势？

答：步进阶梯指令编程常用的有单流程和多流程编程。编程法和步状态分别描述如下。

使用步进阶梯指令编程。很多工业机械动作中，各个动作的特点是按照时间的先后次序遵循一定的顺序规律动作。完善的一套控制系统中，为适应各种控制功能要求，需有手动控制功能、自动控制功能及原点回归功能。自动控制功能中又需点动控制、半自动控制及全自动控制。要实现这些控制功能若用普通顺控程序编程，程序就显得复杂些，同时增加了编程和调试的难度。如果使用步进阶梯指令来编程显然会变得很简单，针对“顺序”逻辑的控制，使用状态软元件 S 配合步进阶梯指令方便地完成工字步进系统的控制。

问 2：如何理解步的状态？

答：对于“步状态”可以这样理解：SCR 指令标志着一个“S”步状态的开始，该步状态内所有逻辑操作的执行与否取决于 S 堆栈的值。如果该段 S 堆栈的值为“1”，该步状态

内所有逻辑操作都执行；如果该段S堆栈的值为“0”，该步状态内所有逻辑操作都不执行。

SCRT转移指令是从现用SCR段向另一个SCR段转移的指令。当执行SCRT转移指令时，会复位当前使用段的S位，并置位指向转移段的S位。

SCRE段指令提供一种退出现用SCR段的格式。

在步进阶梯指令中，SCR和SCRE是成对使用的，在需要转移时就使用SCRT指令。

问3：步进阶梯指令怎样编程？

答：下面举例说明步进阶梯指令的应用方法。

图3-32是十字路口交通灯示意图，图3-33是该十字路口交通灯时序图。使用单流程顺序指令实现的程序如图3-34所示，使用多流程（又称分支与组合）的方法实现的程序如图3-35所示。

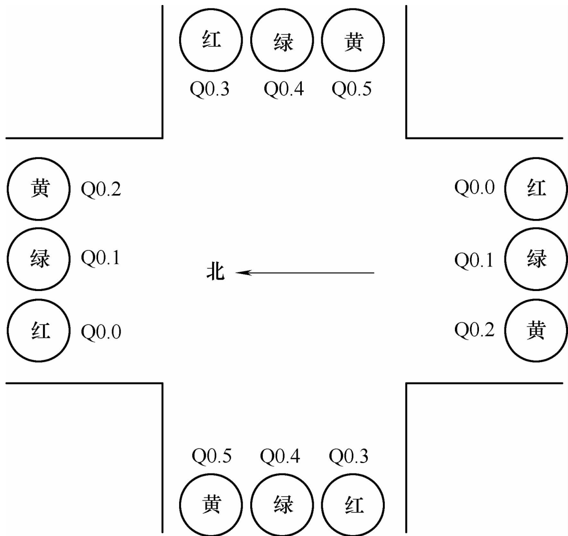


图3-32　十字路口交通灯示意图

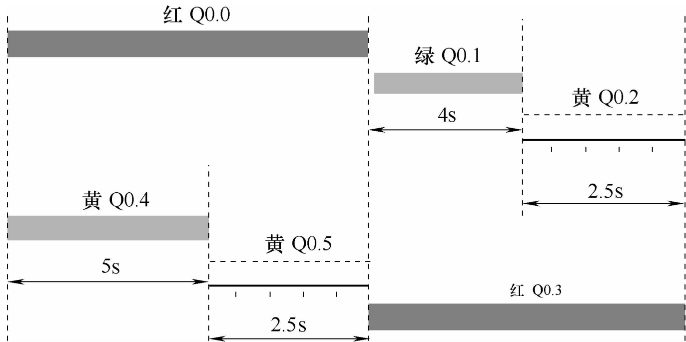


图3-33　十字路口交通灯工艺时序图

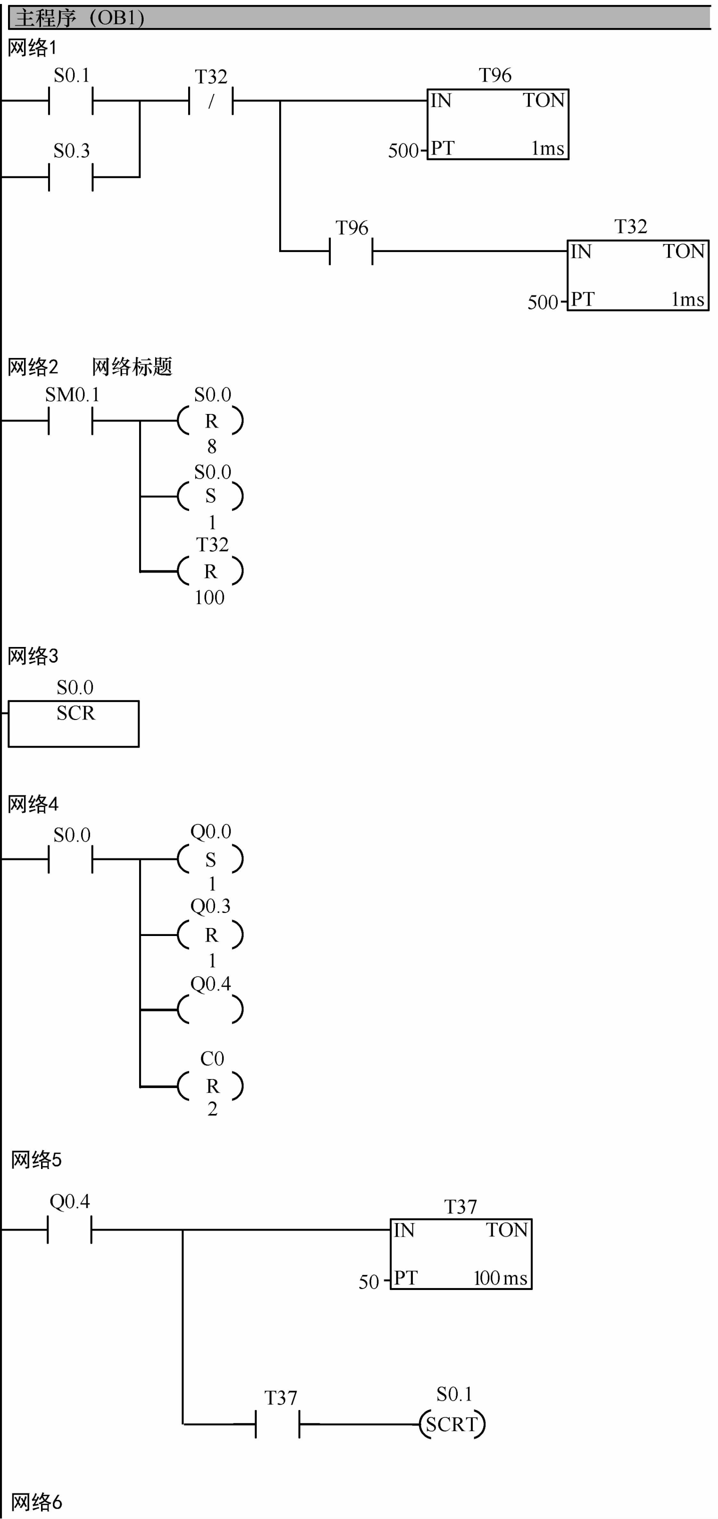


图 3-34 单流程控制程序

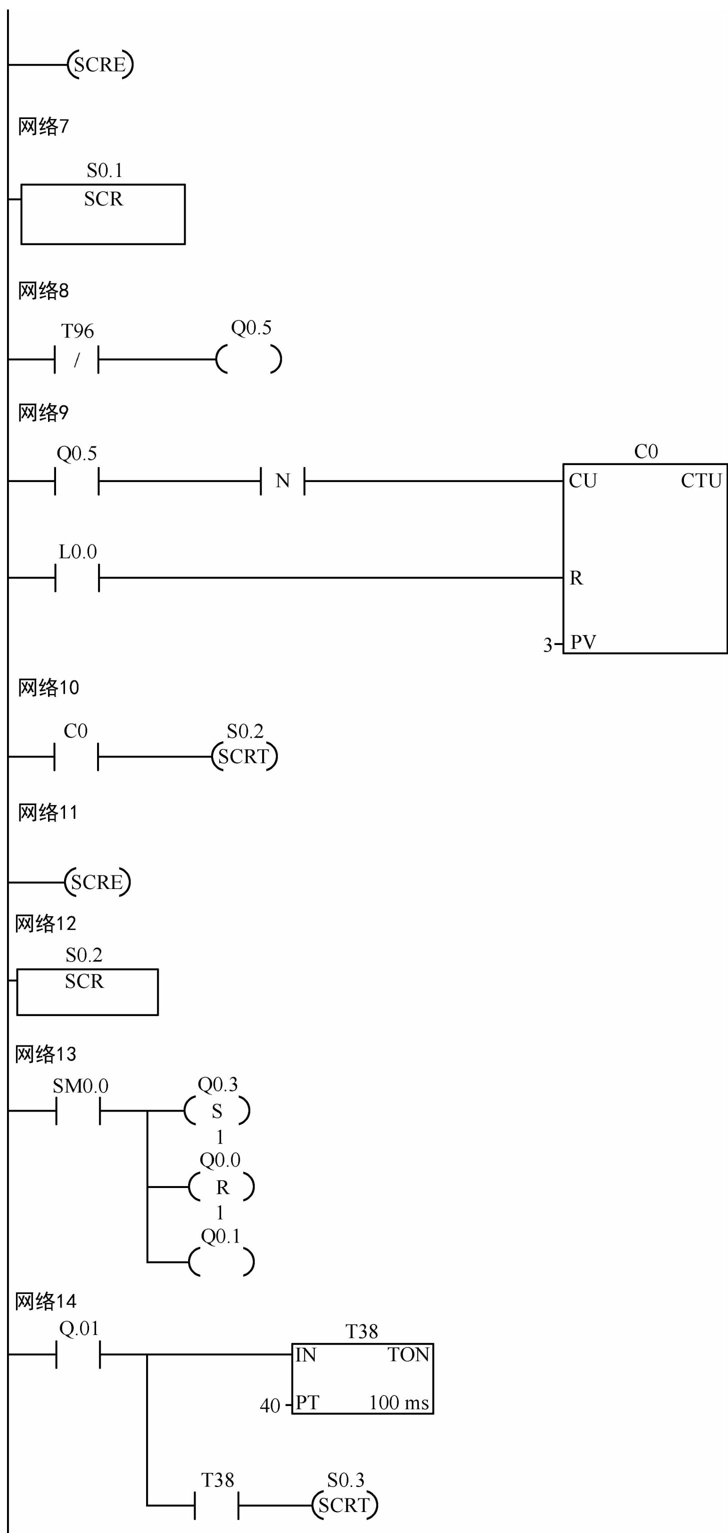


图 3-34 单流程控制程序（续一）

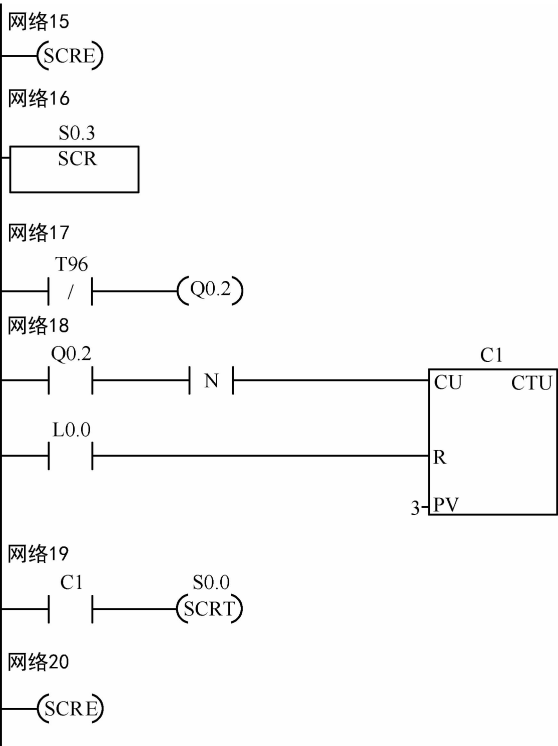


图 3-34　单流程控制程序（续二）

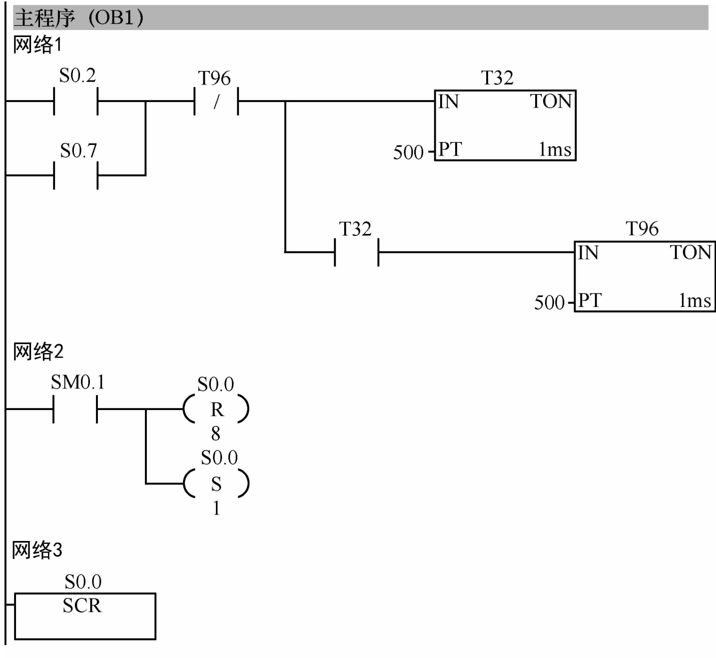


图 3-35　多流程控制程序

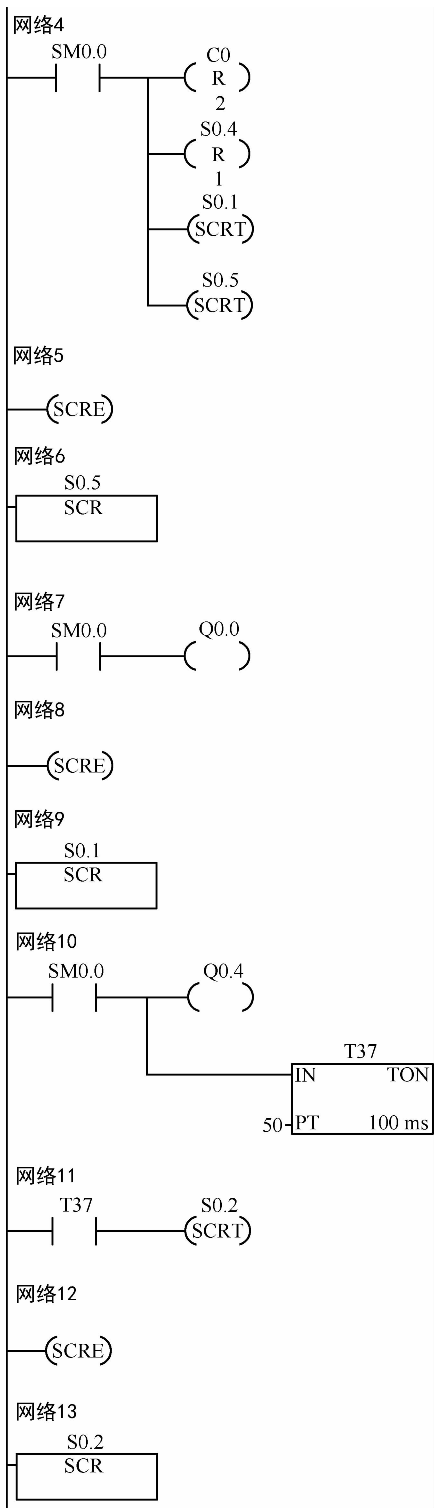


图 3-35　多流程控制程序（续一）

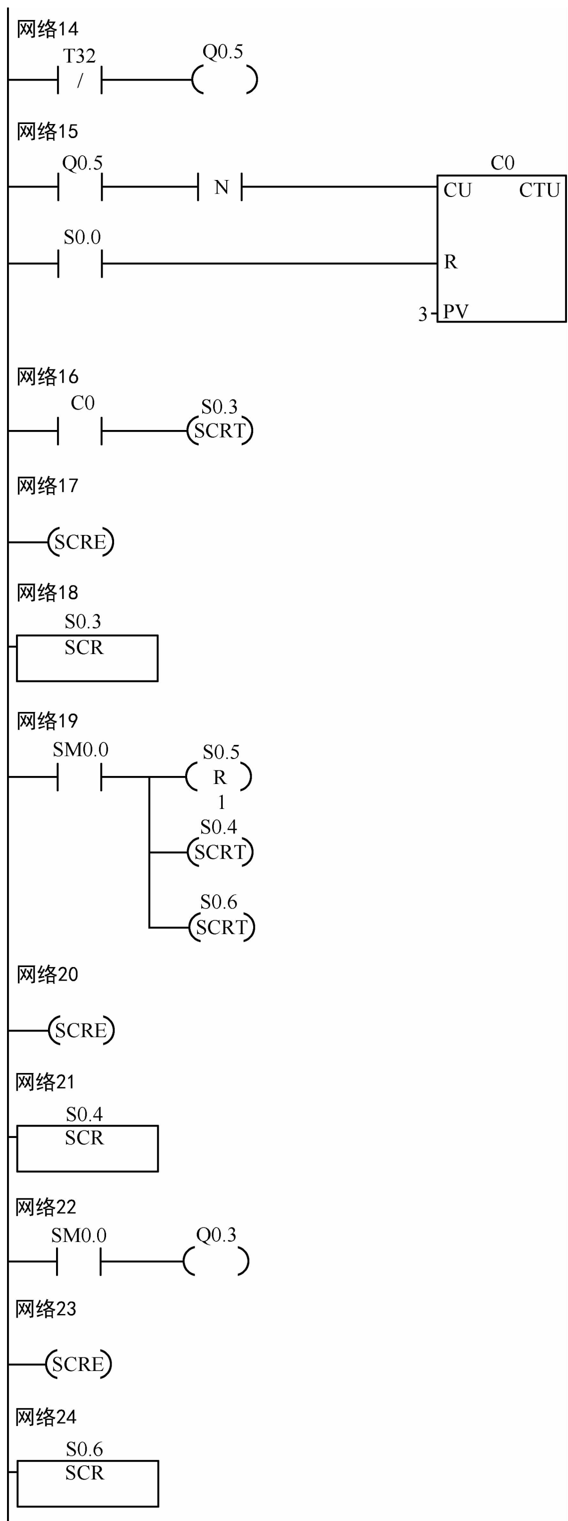


图 3-35　多流程控制程序（续二）

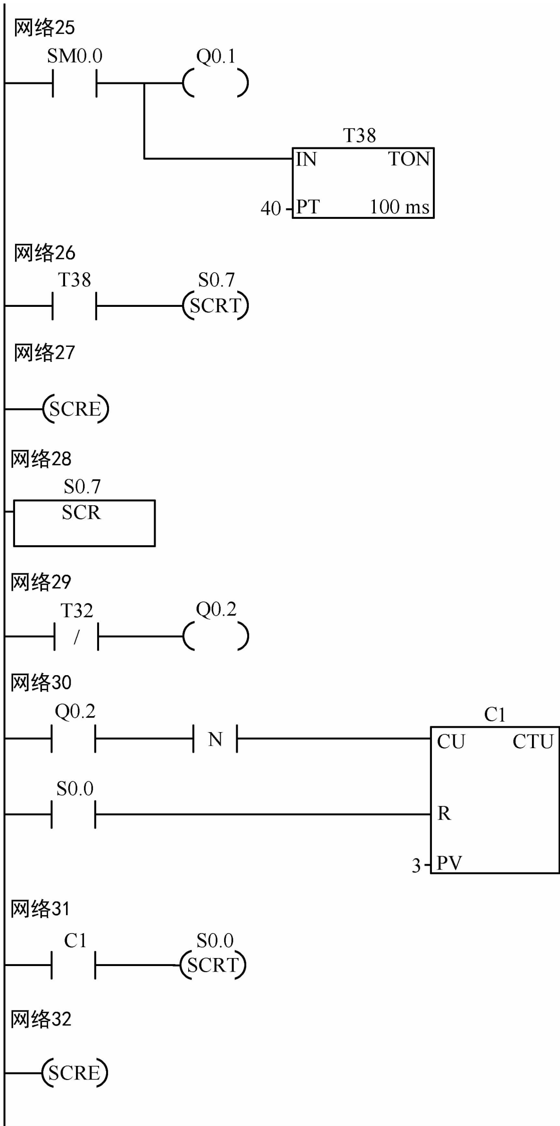


图 3-35 多流程控制程序（续三）

第 4 章　定时器和系统时钟

4.1　TON

问：TON 型定时器的定时规律怎样？ TON 型定时器有哪些？ 如何使用 TON 型定时器？

问 1：TON 型定时器的定时规律怎样？

答：TON 型定时器俗称延时接通定时器，定时规律是当驱动定时器线圈时开始计时，当时间经过值到达预设时间值后，其常开触点导通，常闭触点断开。当在定时过程中驱动信号断开或者有复位命令时，其常开触点及经过值都会复位。

一个完整的定时器包含一个线圈、一对触点、一个设定值和一个经过值，设定值和经过值都是 16 位，数值范围是 0～32767。

定时器预设值设定方法有 3 种：利用常数直接设定、利用变量（IW、QW、VW、MW、SMW、SW、LW、T、C、AC、AIW）设定或用指针（*VD、*LD、*AC）间接设定。

问 2：TON 型定时器有哪些？

答：TON 型定时器见表 4-1，按照时间值单位不同有 100ms、10ms 和 1ms 3 种。

表 4-1　TON 型定时器分类表

	1ms	10ms	100ms
TON	T32、T96	T33～T36，T97～T100	T37～T63，T101～T255

问 3：如何使用 TON 型定时器？

答：TON 型延时接通定时器，I0.0 接通驱动其线圈开始计时，当经过值等于设定值时，对应的触点动作（常开触点闭合，常闭触点断开），这时如果定时器的驱动信号保持（I0.0 保持接通）定时器继续计时，一直计数到经过值达到最大值 32767 时就停止计时，当驱动信号断开（I0.0 断开），其经过值及触点同时会复位，使用 RST 指令也可以使其经过值及触点同时复位。应用程序如图 4-1 所示，操作动作时序图如图 4-2 所示。

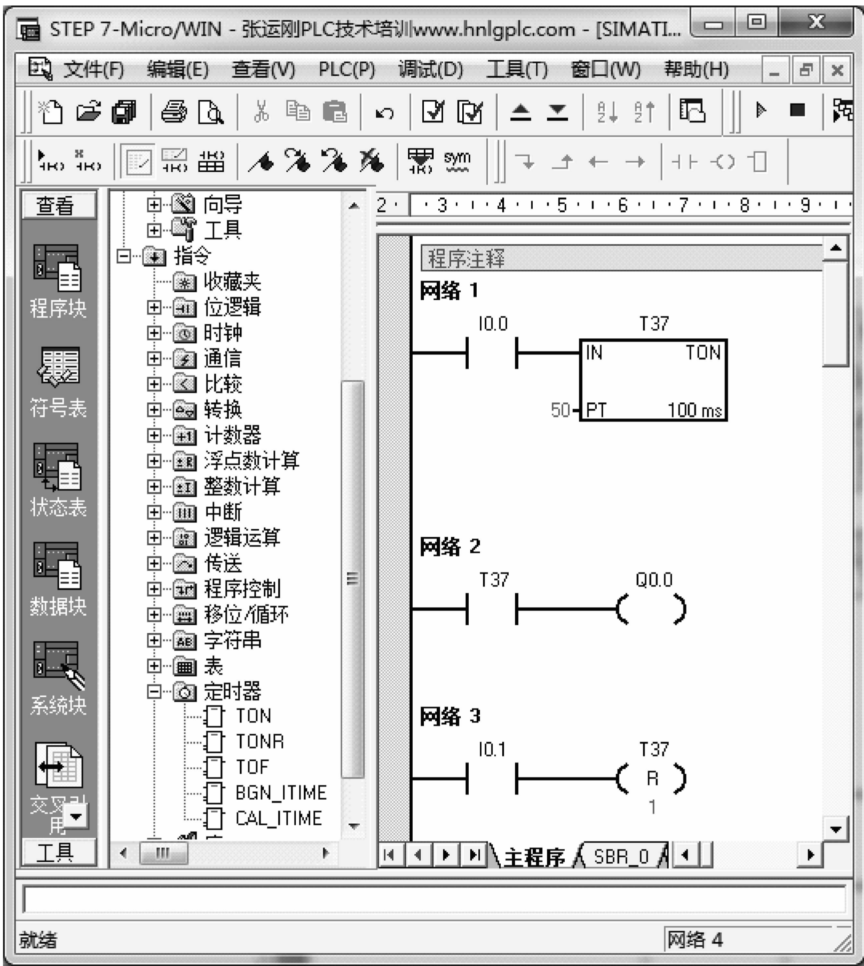


图 4-1　TON 型定时器应用程序

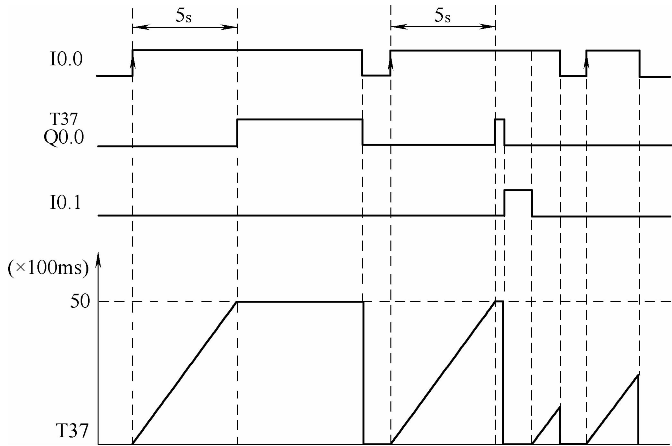


图 4-2　TON 型定时器操作动作时序图

4.2　TOF

问：TOF 型定时器的定时规律怎样？
TOF 型定时器有哪些？
如何使用 TOF 型定时器？

问 1：TOF 型定时器的定时规律怎样？

答：TOF 型定时器俗称延时断开定时器，定时规律是当驱动定时器线圈时其常开触点开始导通，当断开驱动信号开始计时，当时间经过值到达预设时间值后，其常开触点断开，常闭触点接通。当在定时过程中有复位命令时，其常开触点及经过值都会复位。

一个完整的定时器包含一个线圈、一对触点、一个设定值和一个经过值，设定值和经过值都是 16 位，数值范围是 0 ~ 32767。

定时器预设值设定方法有 3 种：利用常数直接设定、利用变量（IW、QW、VW、MW、SMW、SW、LW、T、C、AC、AIW）设定或用指针（ * VD、 * LD、 * AC）间接设定。

问 2：TOF 型定时器有哪些？

答：TOF 型定时器见表 4-2，按照时间值单位不同有 100ms、10ms 和 1ms 三种。

表 4-2　TOF 型定时器分类表

	1ms	10ms	100ms
TOF	T32、T96	T33 ~ T36，T97 ~ T100	T37 ~ T63，T101 ~ T255

问 3：如何使用 TOF 型定时器？

答：TOF 型延时断开定时器，I0.0 接通驱动其线圈其常开触点闭合，当断开驱动信号（I0.0 断开）开始计时，当经过值达到设定值时，对应的触点动作（常开触点断开，常闭触点接通），这时如果定时器的驱动信号继续保持（I0.0 保持接通）定时器也不再计时。在定时过程中如果使用 RST 指令可以使其经过值及触点同时复位。应用程序如图 4-3 所示，操作动作时序图如图 4-4 所示。

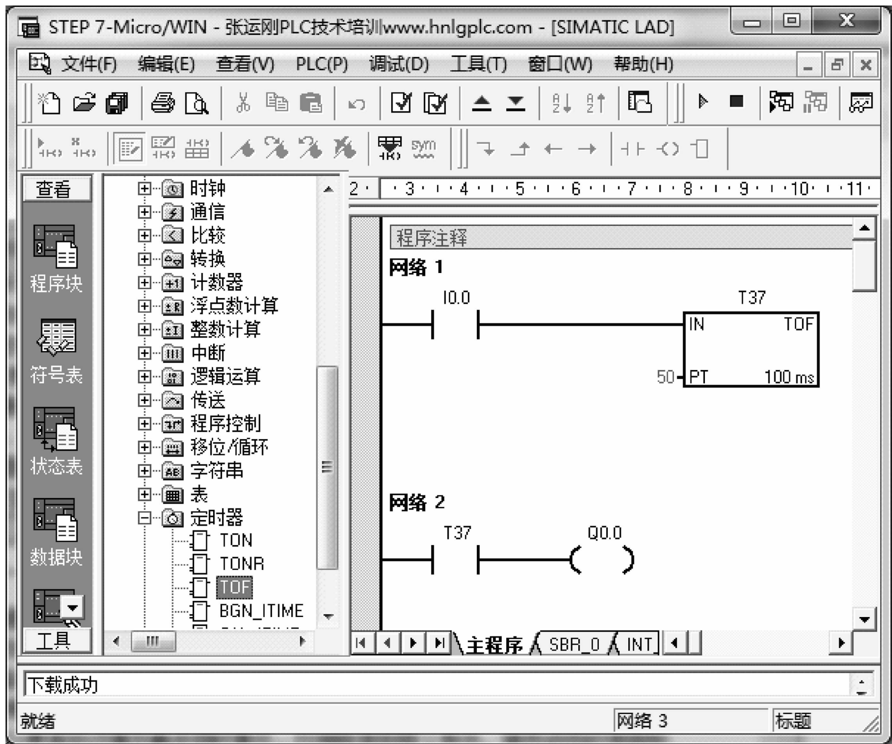


图 4-3　TOF 型定时器应用程序

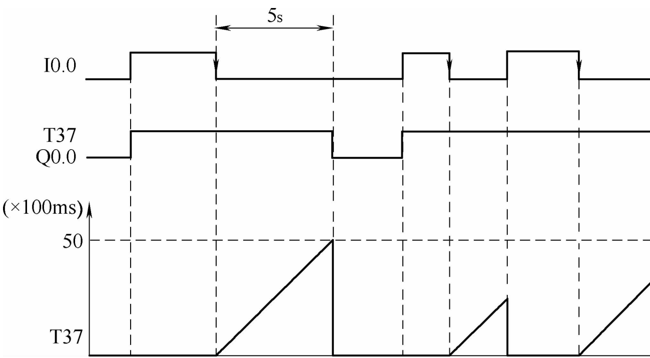


图 4-4　TOF 型定时器操作动作时序图

4.3　TONR

问：TONR 型定时器的定时规律怎样？
TONR 型定时器有哪些？
如何使用 TONR 型定时器？

问 1：TONR 型定时器的定时规律怎样？

答：TONR 型定时器俗称保持型延时接通定时器，定时规律是当驱动定时器线圈时开始计时，当时间经过值到达预设时间值后，其常开触点导通，常闭触点断开。当在定时过程中驱动信号断开其暂停定时，定时器的经过值保持不变。当有复位命令时，其常开触点及经过值都会复位。在定时过程中突然停电，当恢复供电时其经过值保持停电前的状态。

一个完整的定时器包含一个线圈、一对触点、一个设定值和一个经过值，设定值和经过值都是 16 位，数值范围是 0 ~ 32767。

定时器预设值设定方法有 3 种：利用常数直接设定、利用变量（IW、QW、VW、MW、SMW、SW、LW、T、C、AC、AIW）设定或用指针（ * VD、 * LD、 * AC）间接设定。

问 2：TONR 型定时器有哪些？

TONR 型定时器见表 4-3，按照时间值单位不同有 100ms、10ms 和 1ms 3 种。

表 4-3　TONR 型定时器分类表

	1ms	10ms	100ms
TONR	T0、T64	T1 ~ T4，T65 ~ T68	T5 ~ T31，T69 ~ T95

问 3：如何使用 TONR 型定时器？

答：TONR 型保持型延时接通定时器，I0.0 接通驱动其线圈开始计时，当经过值达到设定值时，对应的触点动作（常开触点接通，常闭触点断开），这时如果定时器的驱动信号保持（I0.0 保持接通）定时器继续计时，一直计数到经过值达到最大值 32767 时就停止计时，当驱动信号断开（I0.0 断开），其经过值及触点保持不变，使用 RST 指令可以使其经过值及触点同时复位。应用程序如图 4-5 所示，操作动作时序图如图 4-6 所示。

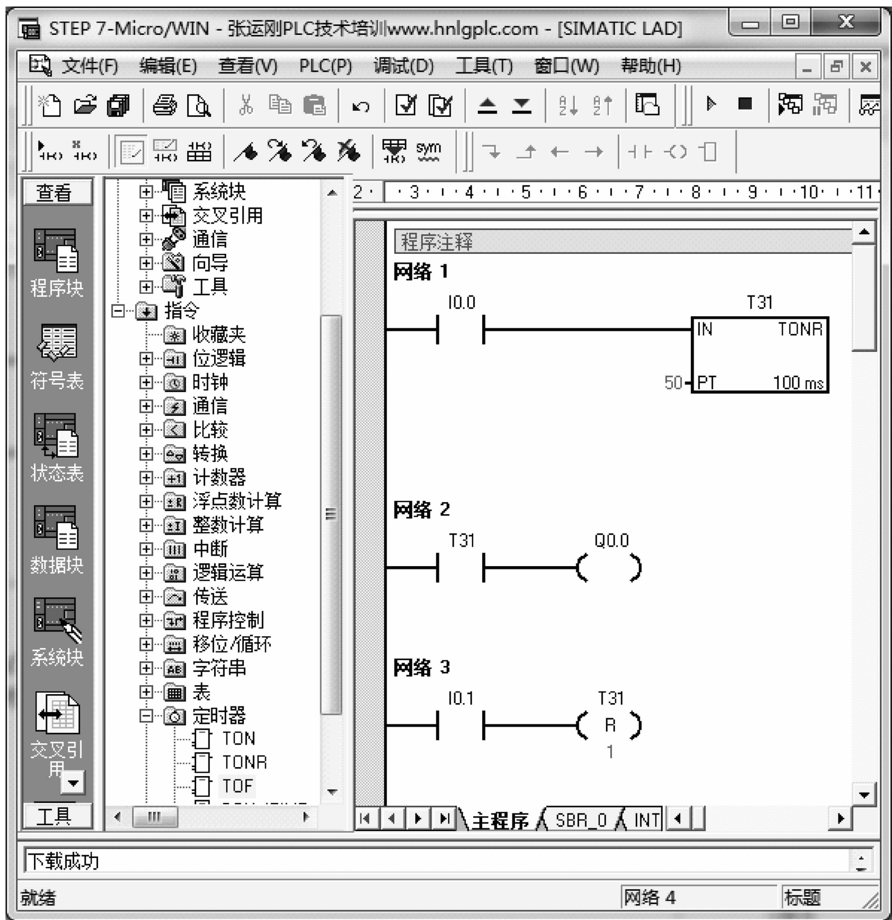


图 4-5　TONR 型定时器应用程序

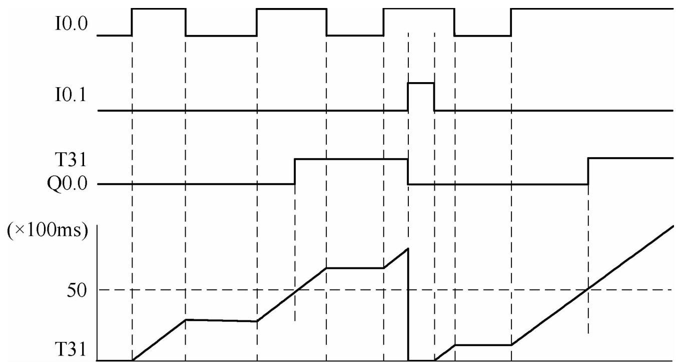


图 4-6　TONR 型定时器操作动作时序图

4.4　BGN_ITIME/CAL_ITIME

问：BGN_ITIME 和 CAL_ITIME 指令基本动作是什么？
　　BGN_ITIME 和 CAL_ITIME 指令如何使用？

问 1：BGN_ITIME 和 CAL_ITIME 指令基本动作是什么？

答：BGN_ITIME 是读取内置 1ms 计数器的当前值，并将该值存储于 OUT。时间单位为 ms，数据长度是 32 位，双字毫秒值的最大计时为 2 的 32 次方 ms（2^{32}ms，大概为 49.7 日）。

CAL_ITIME 是计算当前时间与 IN 所提供时间的时差，将该时差存储于 OUT。时间单位为 ms，数据长度是 32 位，双字毫秒值的最大计时为 2 的 32 次方 ms。

问 2：BGN_ITIME 和 CAL_ITIME 指令如何使用？

答：比如当需要计时 I0.2 接通的总时间，如果接通时间较长达到几天甚至几十天的时长，使用该两条指令就非常方便。实现计算 I0.2 长时间接通时间的程序如图 4-7 所示，总时间保存在 VD500 上，时间单位为 ms。

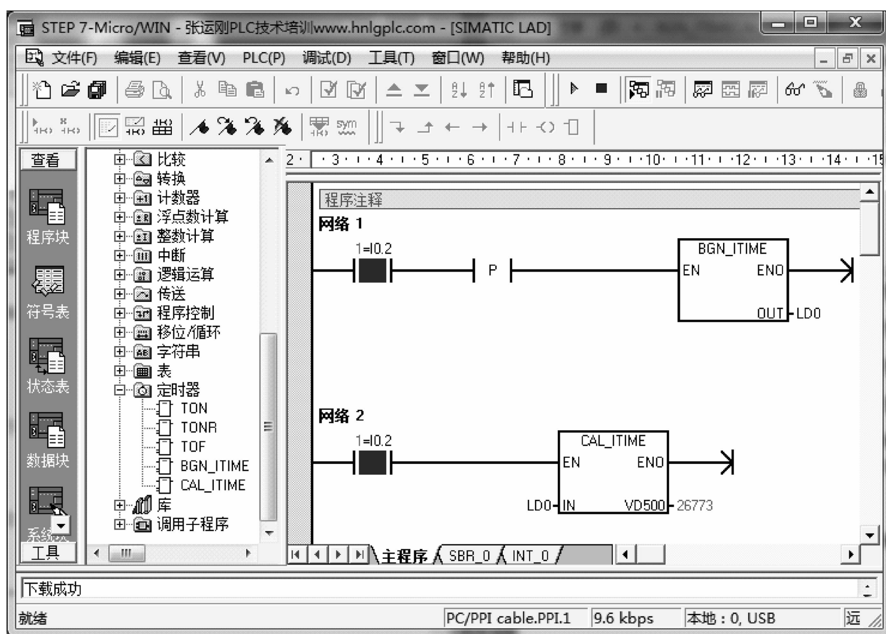


图 4-7　计算 I0.2 长时间接通时间的程序

4.5　READ_RTC/READ_RTCX/SET_RTC/SET_RTCX

问：S7-200 的系统时钟怎样校对时间？如何读取系统时钟？

问：S7-200 的系统时钟怎样校对时间？如何读取系统时钟？

答：S7-200 有两种方式提供时钟功能，一是 CPU 自带的系统时钟或者时钟卡的时钟，二是扩展的系统时钟。读/写自带系统时钟的指令是 READ_RTC/SET_RTC，读/写扩展系统时钟的指令是 READ_RTCX/SET_RTCX。

下面是读/写 CPU 自带的系统时钟的详细描述。

SET_RTC 设置实时时钟指令将预先存放在 T 中的连续 8 个字节的当前时间和日期写入硬件时钟时间缓冲器中，本例具体内容见表 4-4。由于内部没有万年历功能，所以 S7-200 CPU 不会根据日期核实星期几是否正确或日期是否有效，例如 2 月 30 日会被接受。

S7-200 中的当日时间时钟仅使用年份的最后两位数字，因此 2000 年表示为 00。

如果对时钟尝试两个同时存取（非重要错误 0007），SM4.3 标志位会为“1”。

如果 CPU 长时间断电或内存丢失后，当日时间时钟会被初始化为表 4-5 的日期和时间。

表 4-4　T 中 8 个字节时间格式

T 字节偏移地址	如 T 指定为 VB100	说　明	备　注（BCD 码数据类型）
0	VB100	年（0～99）	16#97 代表 97 年
1	VB101	月（1～12）	
2	VB102	日（1～31）	
3	VB103	时（0～23）	
4	VB104	分（0～59）	
5	VB105	秒（0～59）	
6	VB106	00	系统保留为 00
7	VB107	星期（1～7）	1 = 星期日

表 4-5　初始化日期、时间和星期

日　期	时　间	星　期
90 年 01 月 01 日	00:00:00	01（星期日）

READ_RTC 读取实时时钟指令是读取硬件时钟，并将其载入以地址 T 开始的连续 8 个字节的存储器中，T 中连续 8 个字节的具体内容见表 4-4。2096 年之前的闰年均可正确读取。

读/写 CPU 系统时钟的具体程序如图 4-8 所示。

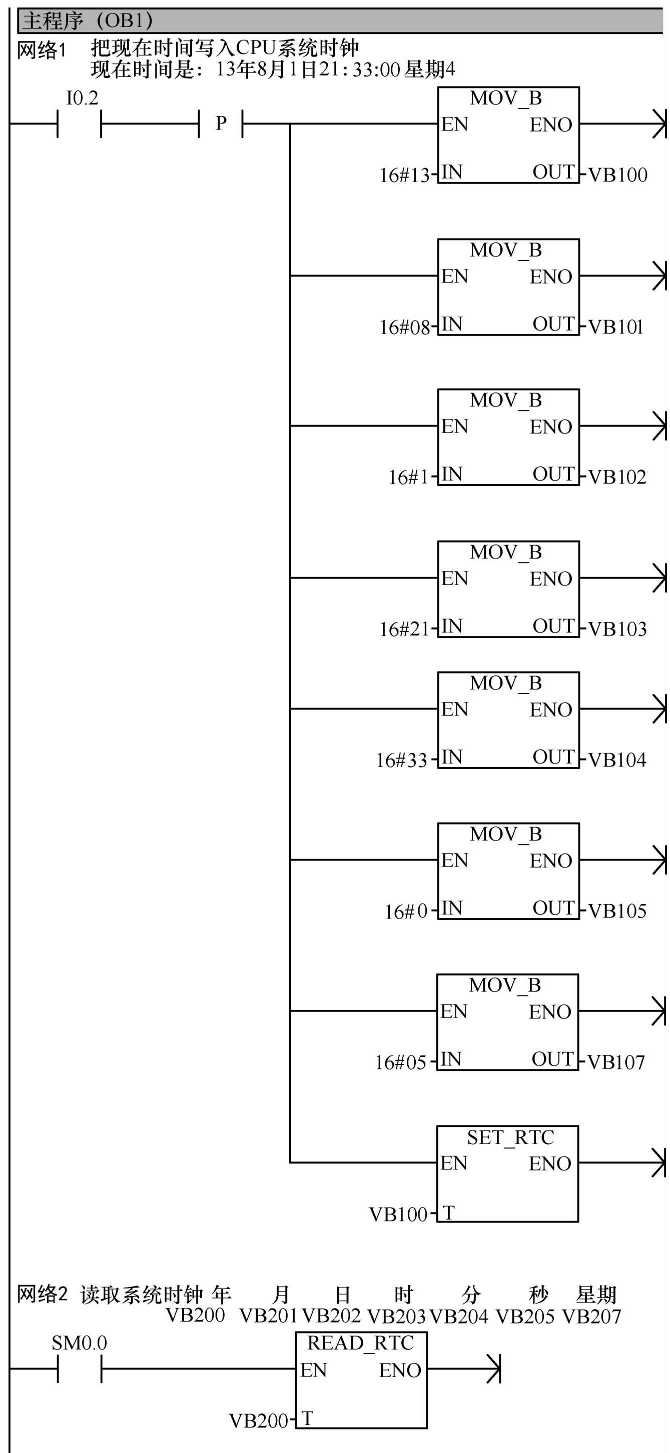


图 4-8　读/写 CPU 系统时钟程序

第 5 章　计　数　器

5.1　CTU

问：CTU 计数规律是什么？怎样探讨 CTU 计数器规律？

问：CTU 计数规律是什么？怎样探讨 CTU 计数器规律？

答： S7-200 有 C0 ~ C255 一共 256 个计数器，默认状态都是停电保持。

一个计数器包含一个线圈、一对触点、一个预设值和一个经过值，设定值和经过值都是 16 位，数值范围是 0 ~ 32767。

计数器预设值设定方法有 3 种：利用常数直接设定、利用变量（IW、QW、VW、MW、SMW、SW、LW、T、C、AC、AIW）设定或用指针（＊VD、＊LD、＊AC）间接设定。

CTU 只加计数器，在计数过程中，当经过值等于或者大于预设值时，其对应的触点动作（常开触点闭合，常闭触点断开），这时如果继续有脉冲输入，计数器继续计数，一直计数到经过值达到最大值 32767 时就停止计数。当使能 R 端或者使用 RST 指令可以复位其经过值和触点。计数器基本动作如图 5-1 所示，基本动作时序图如图 5-2 所示。

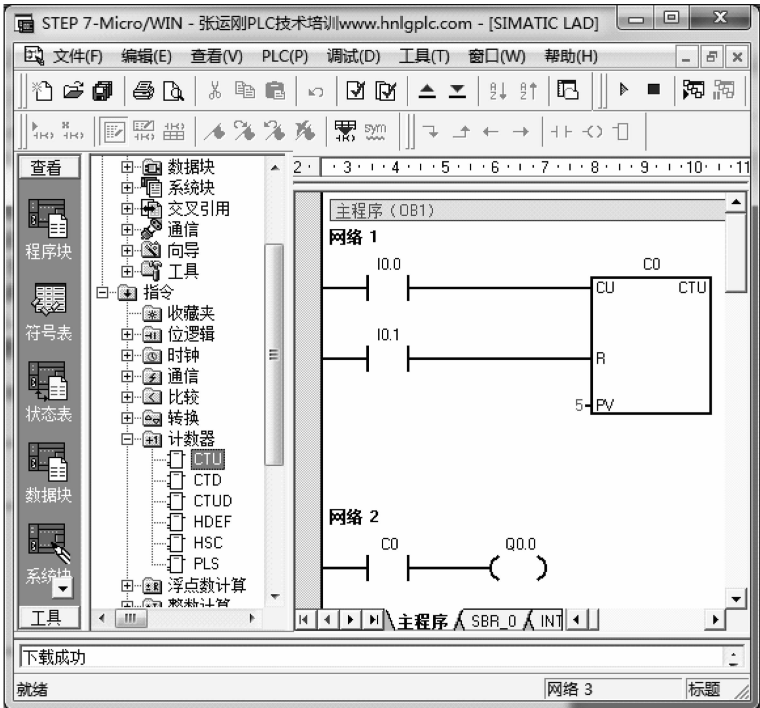


图 5-1　加计数器基本动作

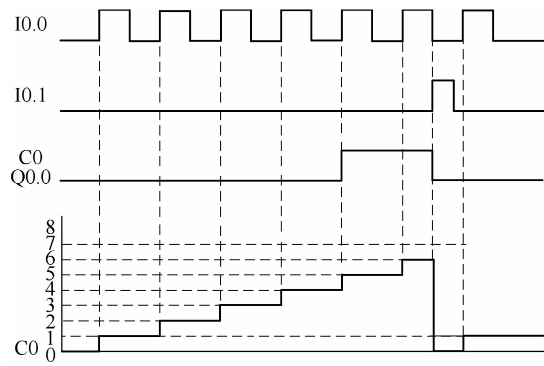


图 5-2　加计数器基本动作时序图

在图 5-3 所示的程序中，加计数器在计数过程中，计数器 C10 的经过值等于设定值，其触点动作，如果这时接通 I0.1，经过值会变为“0”，但 C10 的触点不会断开，当有计数脉冲输入时，触点复位（常开触点断开，常闭触点闭合），同时其经过值会加“1”。

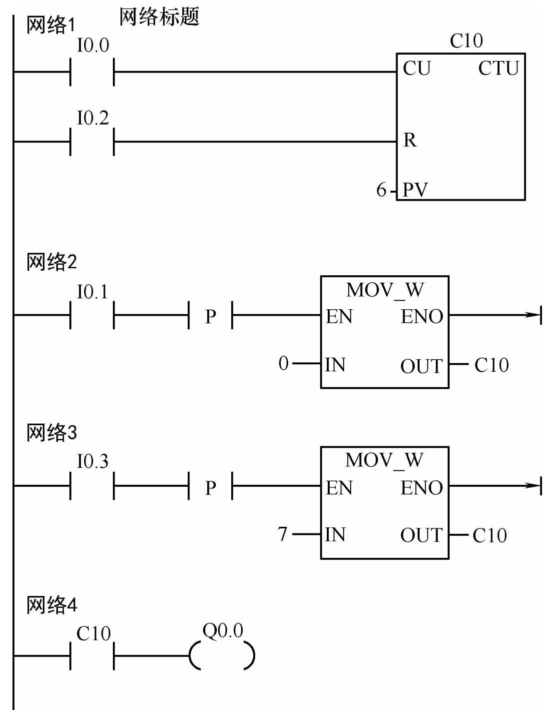


图 5-3　研究加计数器属性程序

在图 5-3 所示的程序中，在计数过程中，计数器 C10 的经过值小于设定值，其触点还没有动作，如果这时接通 I0.3，经过值会变为“7”，但 C10 的触点不会接通，当有计数脉冲输入时，触点动作（常开触点闭合，常闭触点断开），同时其经过值会加“1”。具体动作过程的时序图如图 5-4 所示。

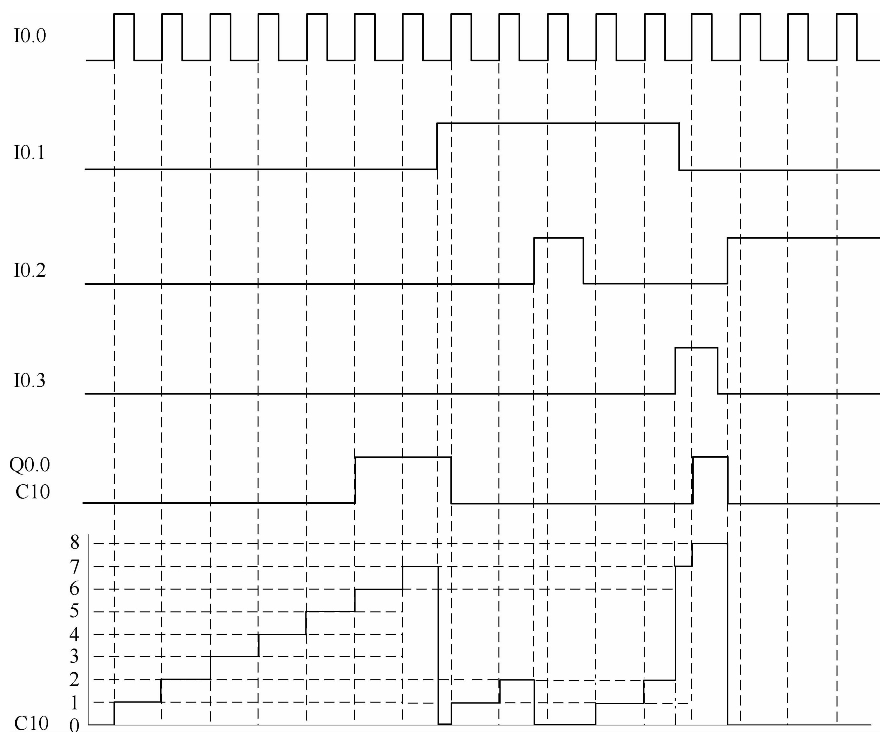


图 5-4 研究加计数器属性时序图

5.2 CTD

问：CTD 计数规律是什么？怎样探讨 CTD 计数器规律？

问：CTD 计数规律是什么？怎样探讨 CTD 计数器规律？

答：S7-200 有 C0～C255 一共 256 个计数器，默认状态都是停电保持。

一个计数器包含一个线圈、一对触点、一个预设值和一个经过值，设定值和经过值都是16 位，数值范围是 0～32767。

计数器预设值设定方法有 3 种：利用常数直接设定、利用变量（IW、QW、VW、MW、SMW、SW、LW、T、C、AC、AIW）设定或用指针（*VD、*LD、*AC）间接设定。

CTD 只减计数器，减计数器在计数过程中，计数器的经过值等于“0”时，其触点动作，常开触点接通，常闭触点断开。当使能 LD 端，会把预设值装载到经过值，其触点会复位。当使用 RST 指令可以复位其经过值和触点。探讨减计数器动作程序如图 5-5 所示，

在图 5-5 所示的程序中，减计数器在计数过程中，计数器 C10 的经过值小于或等于设定值但不为零，其触点还没有动作，如果这时接通 I0.3，经过值会变为“0”，但 C10 的触点不会接通，当有计数脉冲输入时，触点动作（常开触点闭合，常闭触点断开）。其动作如图 5-6 的时序图所示。

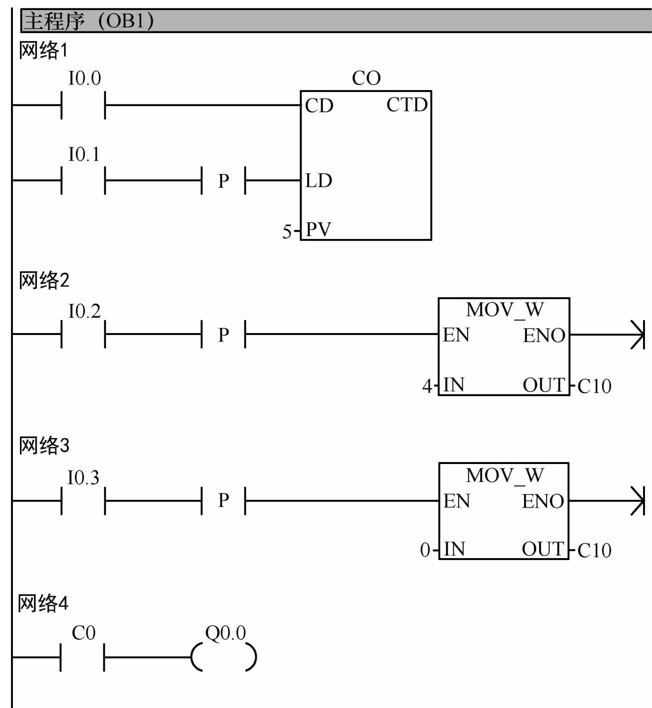


图 5-5　减计数器程序

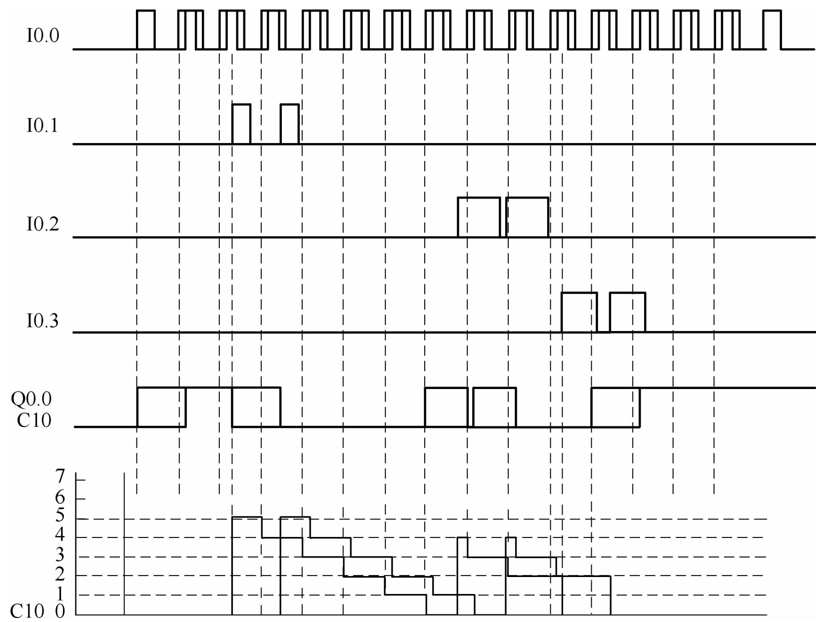


图 5-6　减计数器动作时序图

5.3　CTUD

问：CTUD 计数规律是什么？怎样探讨 CTUD 计数器规律？

问：CTUD 计数规律是什么？怎样探讨 CTUD 计数器规律？

答：S7-200 有 C0～C255 一共 256 个计数器，默认状态都是停电保持。

一个计数器包含一个线圈、一对触点、一个预设值和一个经过值，设定值和经过值都是 16 位，数值范围是 0～32767。CTUD 计数器的数值范围是 -32768～32767。

计数器预设值设定方法有 3 种：利用常数直接设定、利用变量（IW、QW、VW、MW、SMW、SW、LW、T、C、AC、AIW）设定或用指针（＊VD、＊LD、＊AC）间接设定。

CTUD 加减计数器，加减计数器在加计数过程中，计数器的经过值等于或大于预设值时，其触点动作，常开触点接通，常闭触点断开；加减计数器在减计数过程中，计数器的经过值小于预设值时，其触点复位动作，常开触点断开，常闭触点接通。当使能 R 端或者使用 RST 指令可以复位其经过值和触点。探讨加减计数器动作程序如图 5-7 所示。

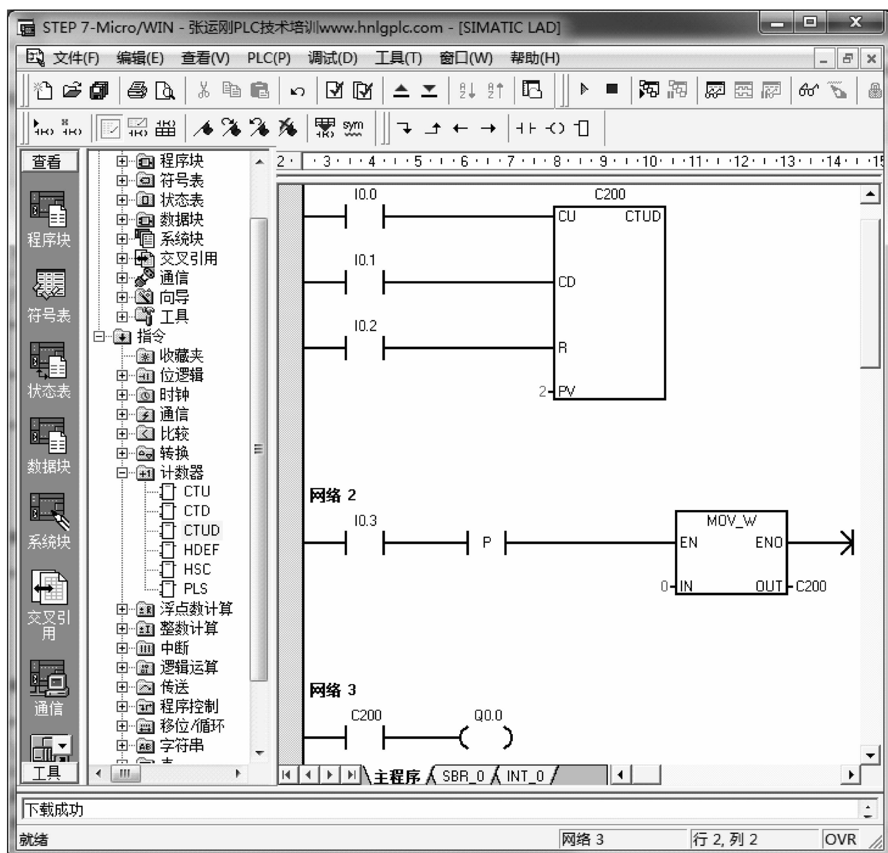


图 5-7　加减计数器程序

在图 5-7 所示程序中，C200 是可加/减计数器，从“CU”输入脉冲是加计数，从

“CD” 输入脉冲则是减计数。可加/减计数器在加计数时，其经过值达到设定值时动作（常开触点接通，常闭触点断开）；在减计数时，其经过值刚好小于设定值，触点复位（常开触点断开，常闭触点闭合）。若用 RST 指令复位，计数器经过值及触点同时都会复位。加减计数器动作时序图如图 5-8 所示。

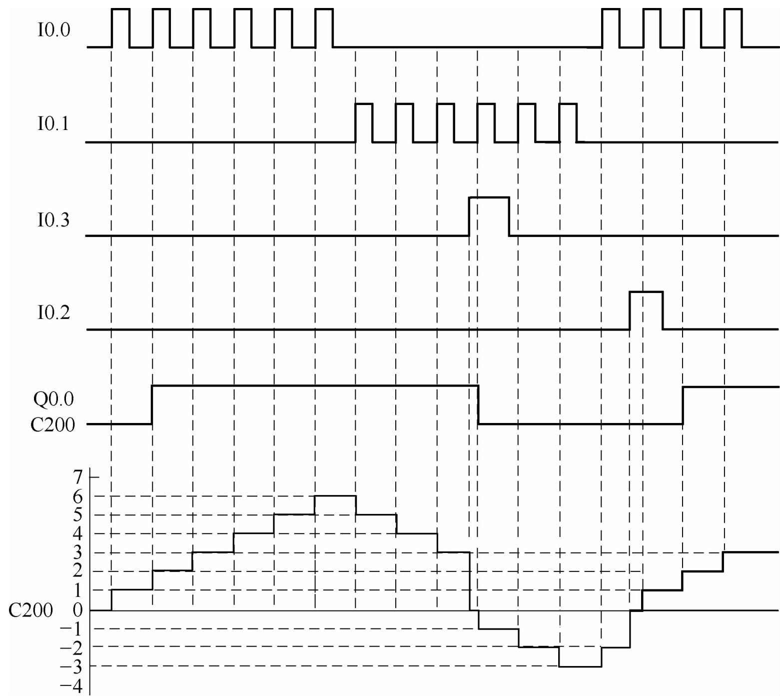


图 5-8　加减计数器动作时序图

第 6 章　传 送 指 令

6.1　MOV_B/W/DW/R

> 问：MOV 指令基本功能是什么？
> MOV 指令样式是怎样的？
> 如何应用 MOV 指令？
> 在应用 MOV 指令时需要注意些什么？

问 1：MOV 指令基本功能是什么？

答：MOV 指令是传送指令，当使能 EN 端子执行 MOV 指令时，会把 IN（称为源）的数值传送到 OUT（称为目标）变量中，源的数值在指令执行前后都不会改变，如果执行指令没有错误，ENO 有能流输出。执行 MOV 指令出错的原因是间接寻址时超出软元件寻址范围。

MOV 指令按照操作数的数据类型不同，分有 MOV_B 字节传送、MOV_W 字传送、MOV_DW 双字传送和 MOV_R 小数传送，使用它们可以实现不同数据类型的传送。

问 2：MOV 指令样式是怎样的？

答：MOV 指令的样式如图 6-1～图 6-4 所示。



图 6-1　MOV_B 字节传送



图 6-2　MOV_W 字传送



图 6-3　MOV_DW 双字传送

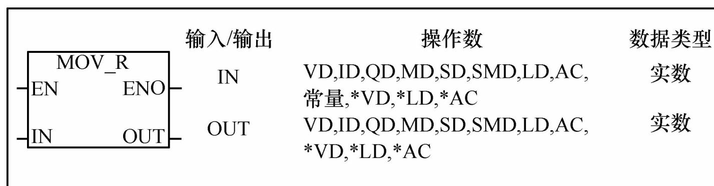


图 6-4　MOV_R 小数传送

问 3：如何应用 MOV 指令？

答：MOV 指令应用描述如下：

MOV_B 字节传送指令应用实例如图 6-5 所示。SM0.1 是开机脉冲，程序中网络 1 的意思是开机初始化 SMB30 和复位 QB0。网络 2 的意思是，当 I0.2 接通时，把常数 7 传送到 VB200，同时把变量 VB201 的值传送到 QB0。网络 3 的意思是，当 I0.3 接通时，把常量 16#0A 传送到 VB201，同时驱动 Q1.0 = 1。

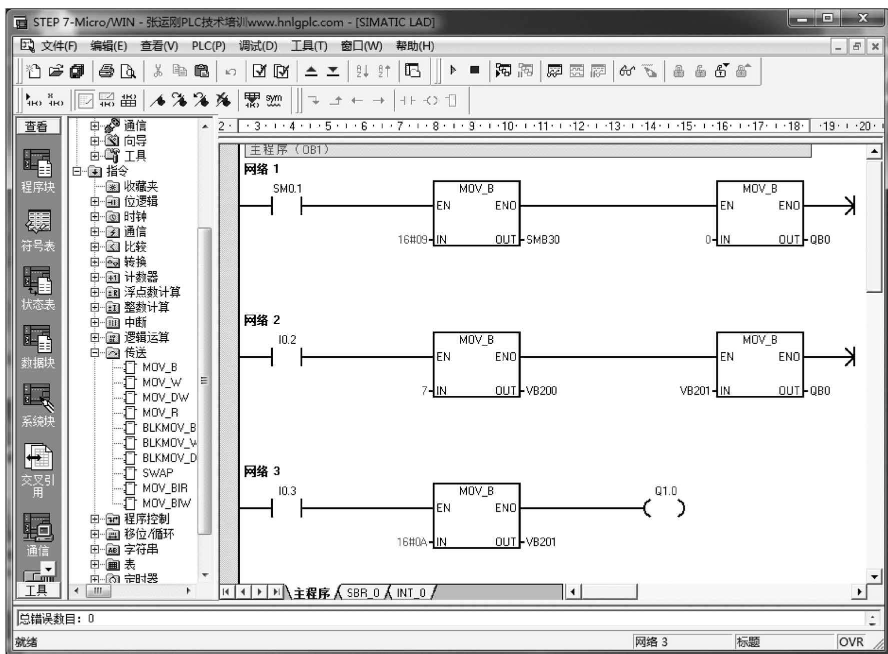


图 6-5　MOV_B 应用程序

MOV_B 字传送指令应用实例 2，控制工艺流程如图 6-6 所示，是一个广场喷泉控制工艺。控制程序如图 6-7 所示，程序流程使用步进阶梯指令控制。

MOV_W 字传送、MOV_DW 双字传送和 MOV_R 小数传送指令应用类似 MOV_B 字节传送指令，只是字传送的操作数是 16 位的字而不是 8 位的字节；MOV_DW 双字传送指令操作数是 32 位；MOV_R 小数传送指令操作数数是小数。

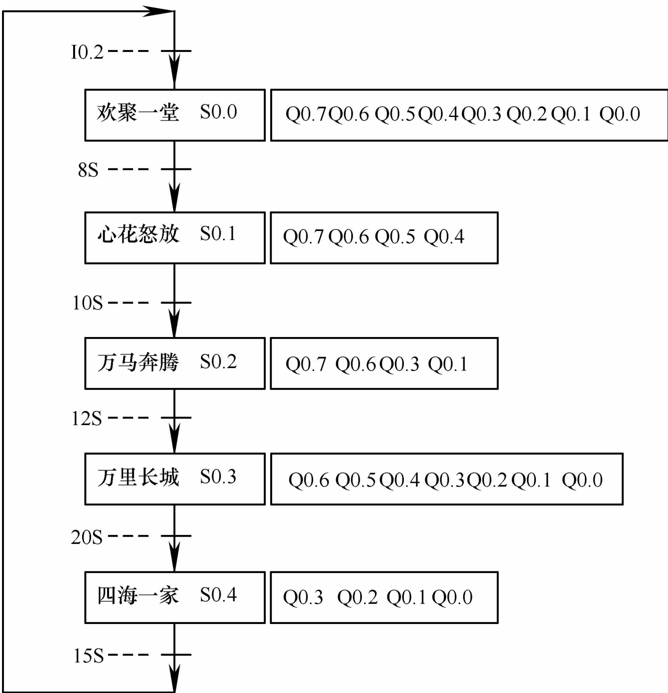


图6-6　广场喷泉控制工艺流程

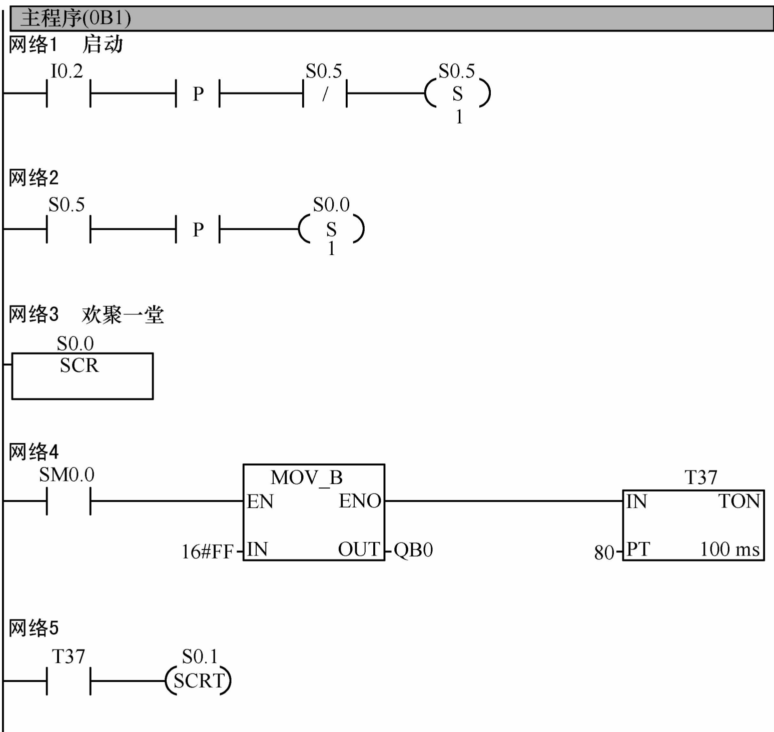


图6-7　MOV_ B应用于喷泉控制程序

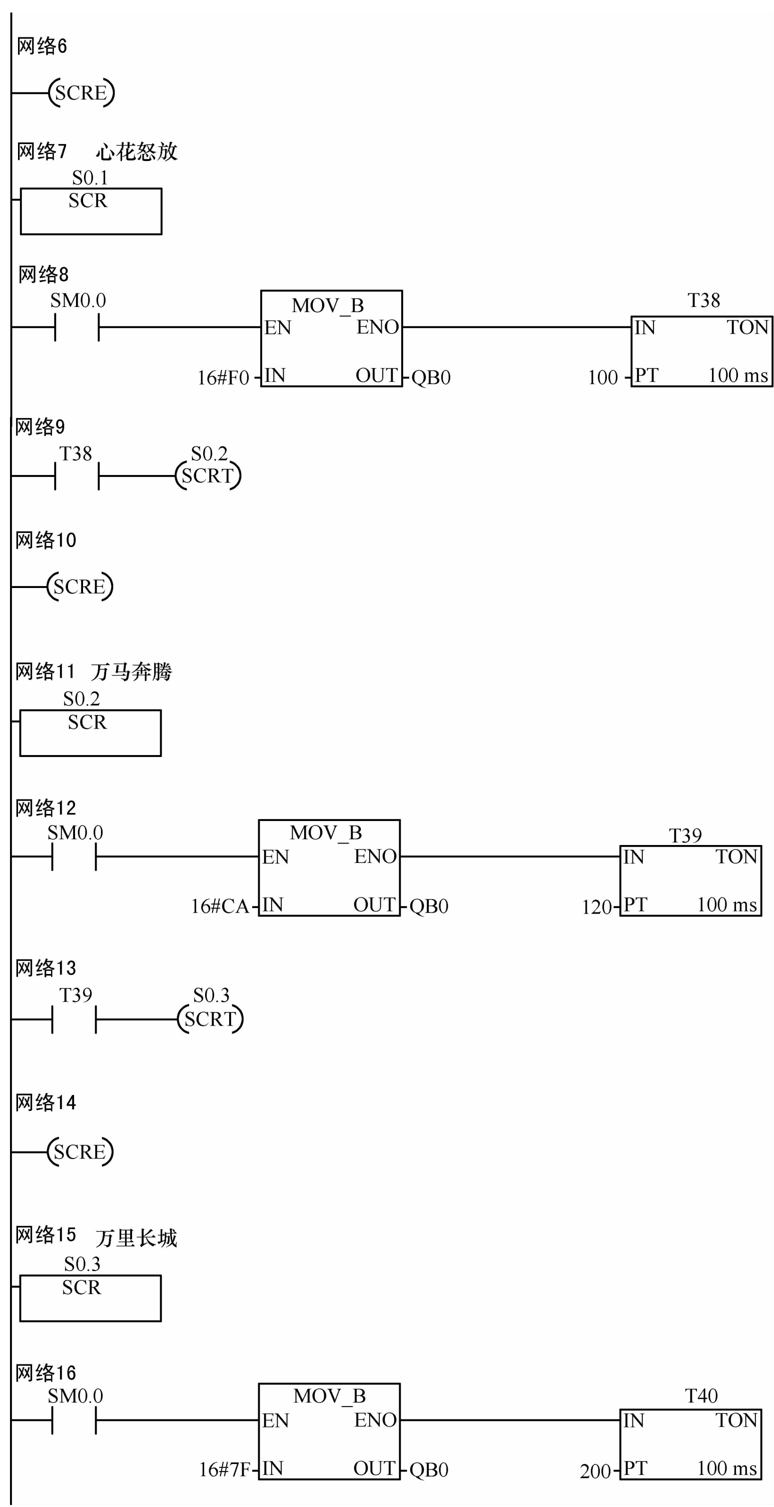


图 6-7　MOV_ B 应用于喷泉控制程序（续一）

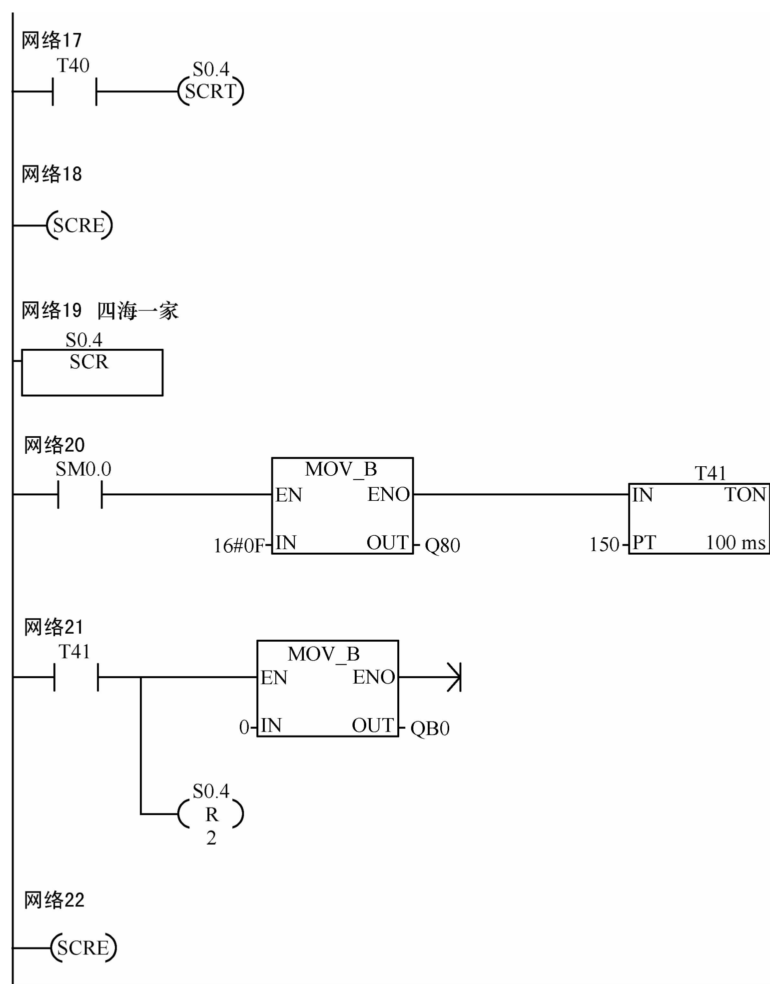


图 6-7　MOV_ B 应用于喷泉控制程序（续二）

问 4：在应用 MOV 指令时需要注意些什么？

答：在应用 MOV 指令时，下面两种情况需要注意的，一是脉冲执行与连续执行有区别，二是当使用间接寻址时需注意不要超出寻址范围。

6.2　BLKMOV_B/W/DW

问：BLKMOV 指令基本功能是什么？ BLKMOV 指令样式是怎样的？ 如何应用 BLKMOV 指令？ 在应用 BLKMOV 指令时需要注意些什么？

问 1：BLKMOV 指令基本功能是什么？

答：BLKMOV 指令是块传送指令，当使能 EN 端子执行 BLKMOV 指令时，会把 IN（称

为源）的整块数值分别传送到 OUT（称为目标）变量中，源和目标块的长度由 N 的数值给定，给定范围是 1 ~ 255，源的数值在指令执行前后都不会改变，如果执行指令没有错误，ENO 有能流输出。执行 BLKMOV 指令出错的原因是间接寻址时超出软元件寻址范围。

BLKMOV 指令按照操作数的数据类型不同，分有 BLKMOV_ B 字节块传送、BLKMOV_W 字块传送和 BLKMOV_DW 双字块传送，使用它们可以实现不同数据类型的块传送。

问 2：BLKMOV 指令样式是怎样的？

答：BLKMOV 块传送指令的样式如图 6-8 ~ 图 6-10 所示。



图 6-8　BLKMOV_B 指令样式



图 6-9　BLKMOV_W 指令样式



图 6-10　BLKMOV_D 指令样式

问 3：如何应用 BLKMOV 指令？

答：当执行 BLKMOV_B 指令时，把 IN 指定开始地址中的若干数据传送到 OUT 指定开始地址的存储器中，具体字节数由 N 指定。

在图 6-11 所示的程序中，当 I0. 0 接通，VB10 开始的 5 个字节（即 VB10 ~ VB14）的数据分别传送到 VB100 开始的 5 个字节中（即 VB100 ~ VB104），如表 6-1 所示。

表 6-1　BLKMOV_B 传送过程

VB10	⟶	VB100	VB13	⟶	VB103
VB11	⟶	VB101	VB14	⟶	VB104
VB12	⟶	VB102			

在使用 BLKMOV_B 指令当 N 中指定的字节数较多时，要注意 IN 和 OUT 中出现重合地址的情况。如图 6-12 所示程序，当 I0.0 接通，VB10 开始的 5 个字节（即 VB10 ~ VB14）的数据分别传送到 VB13 开始的 5 个字节中（即 VB13 ~ VB17），其中 VB13 和 VB14 在 IN 和 OUT 中共用了，如表 6-2 所示。

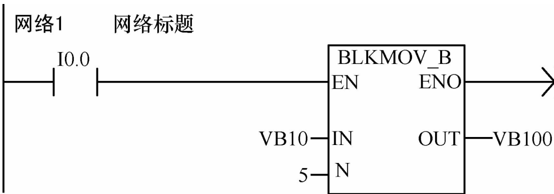


图 6-11　BLKMOV_B 应用程序 1

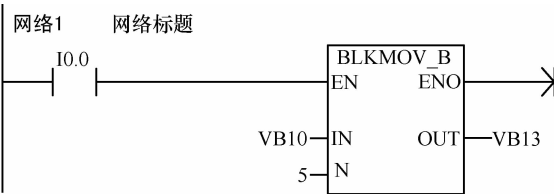


图 6-12　BLKMOV_B 应用程序 2

表 6-2　BLKMOV_B 传送过程

VB10	⟶	VB13	VB13	⟶	VB16
VB11	⟶	VB14	VB14	⟶	VB17
VB12	⟶	VB15			

在使用 BLKMOV_B 指令时注意软元件超出范围，如图 6-13 所示，在 OUT 中从 MB30 开始的 5 个字节（即 MB30 ~ MB34），其中 MB32、MB33、MB34 都是超出范围的软元件（对于 22X CPU），是一种错误操作。

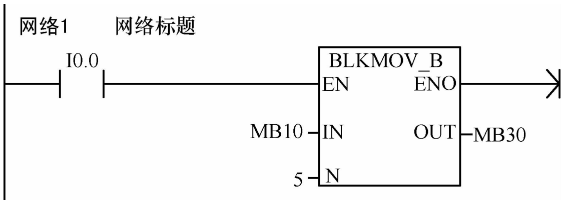


图 6-13　BLKMOV_B 应用程序 3

问 4：在应用 BLKMOV 指令时需要注意些什么？

答：BLKMOV 指令在使用中，除了注意软元件重叠和超出寻址范围外，还需要注意脉冲执行与连续执行指令是有区别的。

BLKMOV_W 字块传送和 BLKMOV_D 双字块传送指令应用类似 BLKMOV_B 字节块传送指令，只是字块传送的操作数是 16 位的字而不是 8 位的字节；MOV_DW 双字块传送指令操作数是 32 位。

6.3　FILL_N

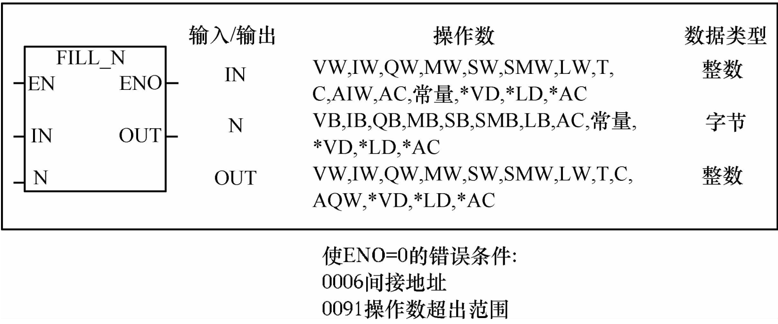
问：FILL_N 指令基本功能是什么？
FILL_N 指令样式是怎样的？
如何应用 FILL_N 指令？
在应用 FILL_N 指令时需要注意些什么？

问 1：FILL_N 指令基本功能是什么？

答： FILL_N 指令是一点多送指令，当使能 EN 端子执行 FILL_N 指令时，会把 IN（称为源）的数值分别传送到 OUT（称为目标）变量中，目标块的长度由 N 的数值给定，给定范围是 1 ~255，源的数值在指令执行前后都不会改变，如果执行指令没有错误，ENO 有能流输出。执行 FILL_N 指令出错的原因是间接寻址时超出软元件寻址范围。

问 2：FILL_N 指令样式是怎样的？

答： FILL_N 指令样式如图 6-14 所示。



输入/输出	操作数	数据类型
IN	VW,IW,QW,MW,SW,SMW,LW,T,C,AIW,AC,常量,*VD,*LD,*AC	整数
N	VB,IB,QB,MB,SB,SMB,LB,AC,常量,*VD,*LD,*AC	字节
OUT	VW,IW,QW,MW,SW,SMW,LW,T,C,AQW,*VD,*LD,*AC	整数

使ENO=0的错误条件:
0006间接地址
0091操作数超出范围

图 6-14　FILL_N 指令样式

问 3：如何应用 FILL_N 指令？

答： 如果要实现图 6-15 所示程序的功能，可以使用 FILL_N 指令来实现，如图 6-16 所示程序。图 6-15 或图 6-16 所示程序中如果 VW10 =0，在 I0. 0 的上升沿执行 MOV 指令或者 FILL_N 指令使 VW100 ~ VW106 的数值都为 0，相当于复位或者说清零。

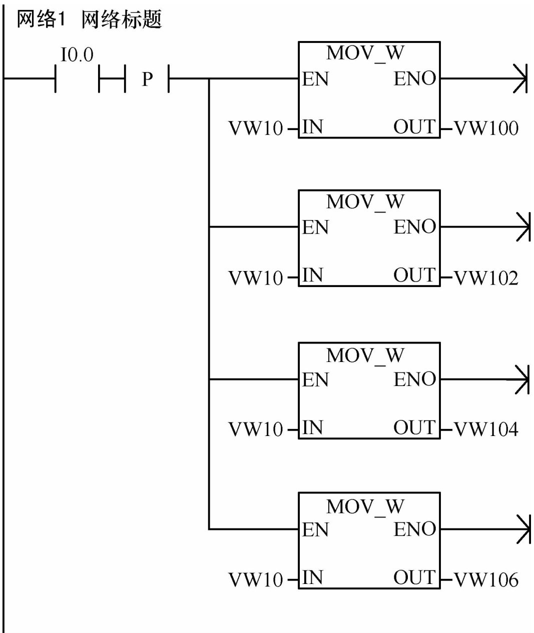


图 6-15　使用 FILL_N 指令

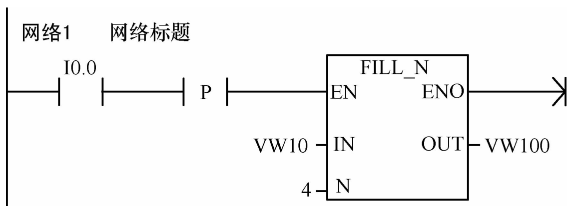


图 6-16　使用 FILL_ N 指令

问 4：在应用 FILL_ N 指令时需要注意些什么？

答：在应用 FILL_ N 指令时，下面两种情况需要注意的，一是脉冲执行与连续执行有区别，二是当使用间接寻址时需注意不要超出寻址范围。

6.4　SWAP

问：SWAP 指令基本功能是什么？
SWAP 指令样式是怎样的？
如何应用 SWAP 指令？
在应用 SWAP 指令时需要注意些什么？

问 1：SWAP 指令基本功能是什么？

答：SWAP 指令是字节交换指令，当使能 EN 端子执行 SWAP 指令时，会把 IN（称为源）的字的上下字节数据互相交换，如果执行指令没有错误，ENO 有能流输出。执行 SWAP 指令出错的原因是间接寻址时超出软元件寻址范围。

问 2：SWAP 指令样式是怎样的？

答：SWAP 指令样式如图 6-17 所示。



图 6-17　SWAP 指令样式

问 3：如何应用 SWAP 指令？

答：当执行 SWAP 指令时，IN 中指定字的上下字节的内容互相交换。在图 6-18 所示的程序中，当 VW10 = 16#2033 时，接通 I0. 0 后结果变为 VW10 = 16#3320；当 VW10 = 16#3120 时，接通 I0. 0 后结果变为 VW10 = 16#2031。

在使用 SWAP 字节交换指令时，要注意使用脉冲型，不然很可能得不到需要的结果，除非确保驱动信号只接通一个扫描周期的次数（或奇数个扫描周期的次数），程序如图 6-19

所示。

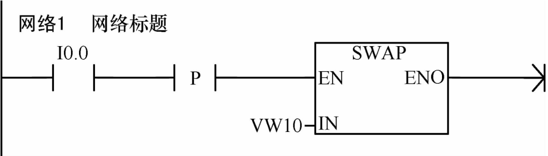


图 6-18　SWAP 指令应用 1

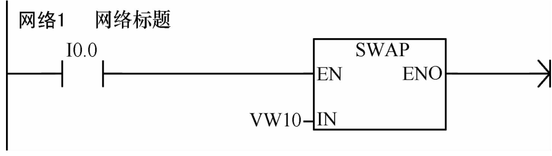


图 6-19　SWAP 指令应用 2

问 4：在应用 SWAP 指令时需要注意些什么？

答：在应用 SWAP 指令时，下面两种情况需要注意的，一是脉冲执行与连续执行有区别，二是当使用间接寻址时需注意不要超出寻址范围。

6.5　INV_B/W/DW

> 问：INV 指令基本功能是什么？
> INV 指令样式是怎样的？
> 如何应用 INV 指令？
> 在应用 INV 指令时需要注意些什么？

问 1：INV 指令基本功能是什么？

答：INV 指令是取反传送指令，当使能 EN 端子执行 INV 指令时，会把 IN（称为源）的数值取反传送到 OUT（称为目标）变量中，源的数值在指令执行前后都不会改变，如果执行指令没有错误，ENO 有能流输出。执行 INV 指令出错的原因是间接寻址时超出软元件寻址范围。

INV 指令按照操作数的数据类型不同，分有 INV_B 字节取反传送、INV_W 字取反传送和 INV_DW 双字取反传送，使用它们可以实现不同数据类型的取反传送。

问 2：INV 指令样式是怎样的？

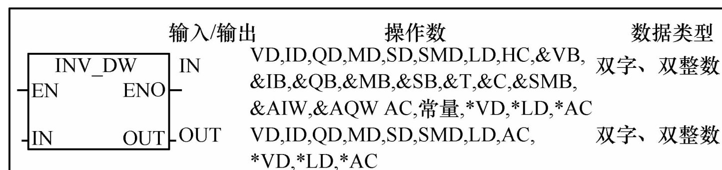
答：INV 取反传送指令的样式如图 6-20 ~ 图 6-22 所示。

INV_B (EN, ENO, IN, OUT)	输入/输出	操作数	数据类型
	IN	VB,IB,QB,MB,SB,SMB,LB,AC,常量,*VD,*LD,*AC	字节
	OUT	VB,IB,QB,MB,SB,SMB,LB,AC,*VD,*LD,*AC	字节

图 6-20　INV_B 字节取反传送

INV_W (EN, ENO, IN, OUT)	输入/输出	操作数	数据类型
	IN	VW,IW,QW,MW,SW,SMW,LW,T,C,AIW,常量,AC,*VD,*AC,*LD	字、整数
	OUT	VW,T,C,IW,QW,SW,MW,SMW,LM,AC,AQW,*VD,*AC,*LD	字、整数

图 6-21　INV_W 字取反传送



输入/输出	操作数	数据类型
IN	VD,ID,QD,MD,SD,SMD,LD,HC,&VB,&IB,&QB,&MB,&SB,&T,&C,&SMB,&AIW,&AQW AC,常量,*VD,*LD,*AC	双字、双整数
OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	双字、双整数

图 6-22　INV_DW 双字取反传送

问 3：如何应用 INV 指令？

答：实现图 6-23 所示程序的功能，如果使用 INV_B 字节取反传送指令很方便就可以实现，如图 6-24 所示。

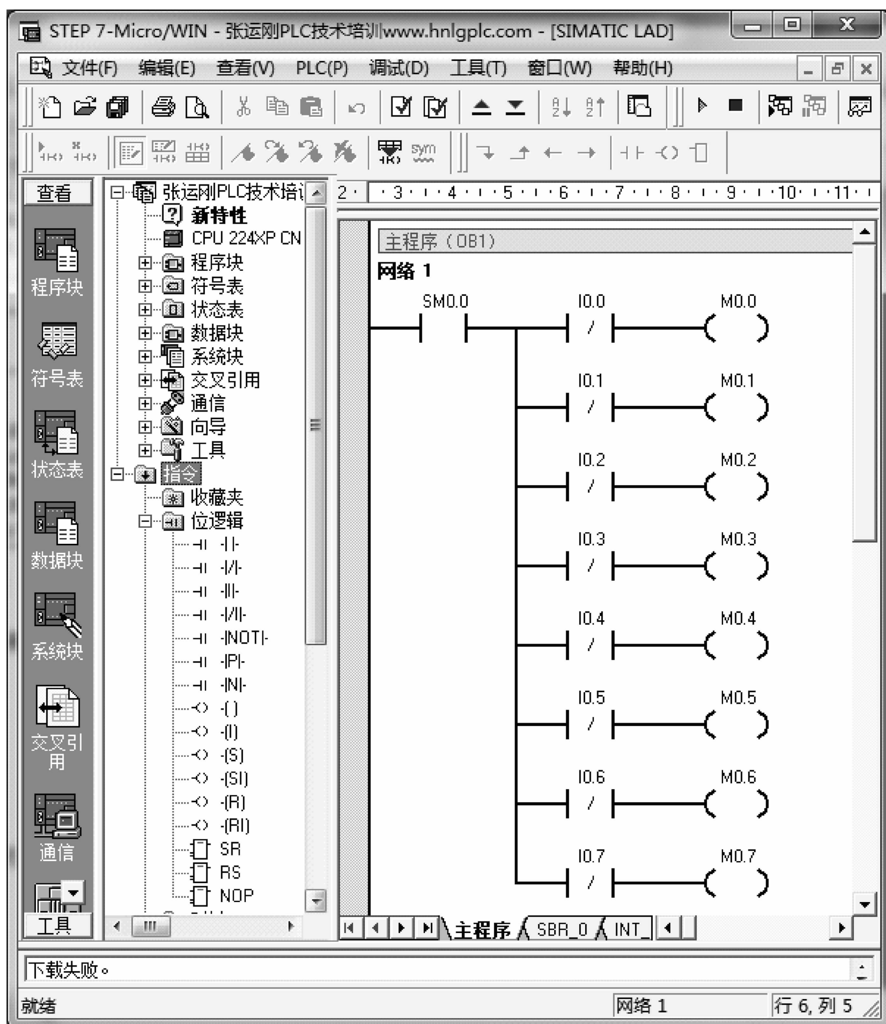


图 6-23　取反触点应用

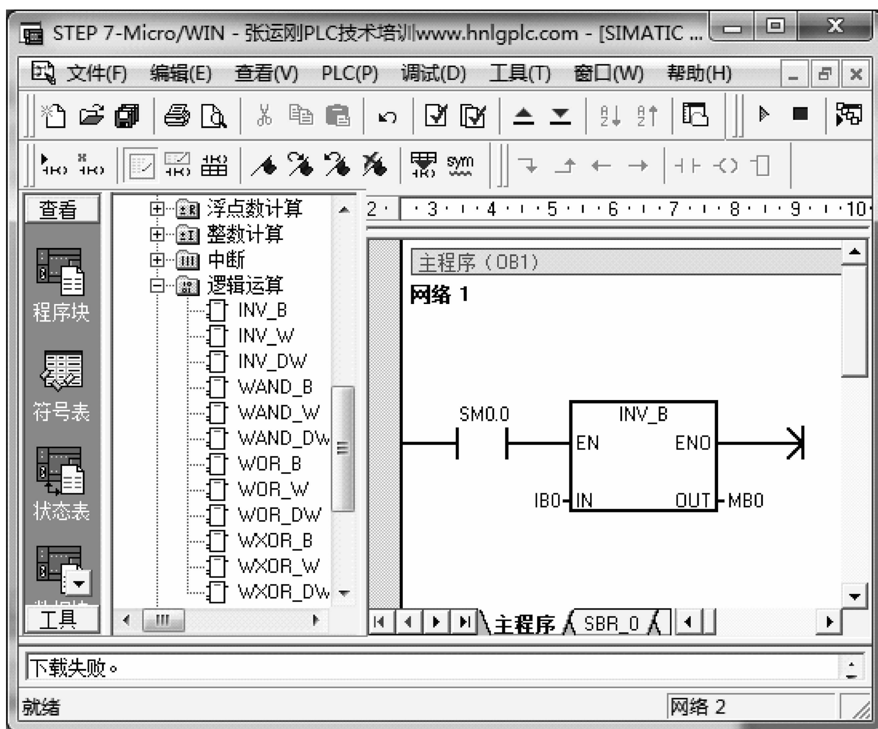


图 6-24　INV_B 应用程序 1

在使用 INV_B 字节取反传送指令时，要注意使用脉冲型，不然很可能得不到需要的结果，除非确保驱动信号只接通一个扫描周期的次数（或奇数个扫描周期的次数），程序如图 6-25 所示。

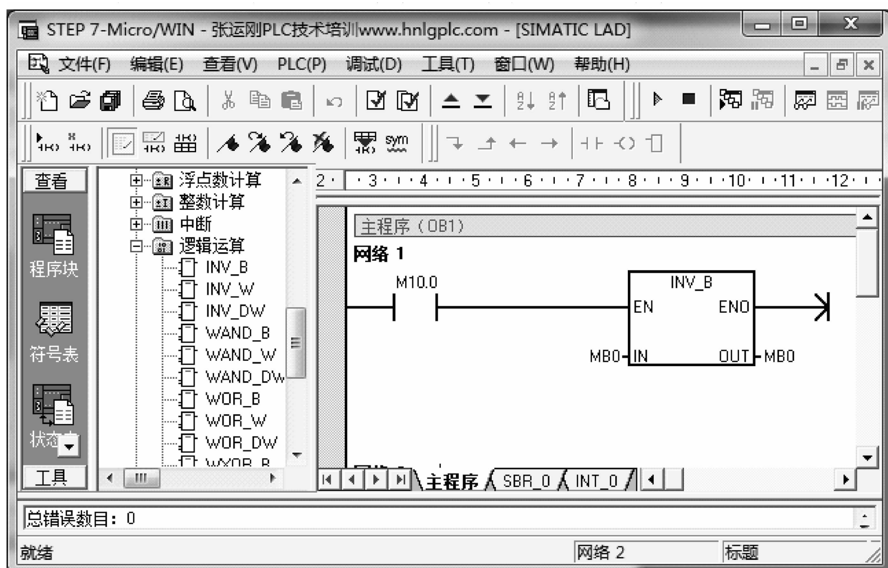


图 6-25　INV_B 应用程序 2

INV_W 字取反传送、INV_DW 双字取反传送指令应用类似 INV_B 字节取反传送指令，只是字取反传送的操作数是 16 位的字而不是 8 位的字节；INV_DW 双字取反传送指令操作数是 32 位。

问 4：在应用 INV 指令时需要注意些什么？

答：在应用 INV 指令时，下面两种情况需要注意的，一是脉冲执行与连续执行有区别，二是当使用间接寻址时需注意不要超出寻址范围。

6.6 MOV_BIR/MOV_BIW

问：MOV_BIR 和 MOV_BIW 指令基本功能是什么？
MOV_BIR 和 MOV_BIW 指令样式是怎样的？
如何应用 MOV_BIR 和 MOV_BIW 指令？
在应用 MOV_BIR 和 MOV_BIW 指令时需要注意些什么？

问 1：MOV_BIR 和 MOV_BIW 指令基本功能是什么？

答：MOV_BIR 指令是立即读指令,当使能 EN 端子执行 MOV_BIR 指令时，会把 IN（称为源）的数值读到 OUT（称为目标）变量中，源的数值是物理输入字节，执行该指令时相应输入点的输入映像区并未更新。如果执行指令没有错误，ENO 有能流输出。执行 MOV_BIR 指令出错的原因是间接寻址时超出软元件寻址范围。

MOV_BIR 的源一定是输入字节,指向其他软元件是错误的。

MOV_BIW 指令是立即写指令，当使能 EN 端子执行 MOV_BIW 指令时，会把 IN（称为源）的数值写到 OUT（称为目标）变量中，目标的数值是物理输出字节，执行该指令时相应输出点的输出映像区并未更新。如果执行指令没有错误，ENO 有能流输出。执行 MOV_BIW 指令出错的原因是间接寻址时超出软元件寻址范围。

MOV_BIW 的目标一定是输出字节，指向其他软元件也是错误的。

问 2：MOV_BIR 和 MOV_BIW 指令样式是怎样的？

答：MOV_BIR 指令样式如图 6-26 所示，MOV_BIW 指令样式如图 6-27 所示。



图 6-26 MOV_BIR 指令样式

MOV_BIW	输入/输出	操作数	数据类型
EN ENO	IN	VB,IB,QB,MB,SB,SMB,LB, AC,常量,*VD,*AC,*LD	字节
IN OUT	OUT	QB,*VD,*LD,*AC	字节

使ENO=0的错误条件:
0006间接地址

图 6-27　MOV_BIW 指令样式

问 3：如何应用 MOV_BIR 和 MOV_BIW 指令？

答：PLC 在处理输入和输出的时候，是按照扫描方式，每个扫描周期都成批输入刷新和成批输出刷新。在执行网络段 1 指令前把 I 输入的状态成批刷新到输入映像区，在执行 END 前在所有指令执行完后，把 Q 输出映像区的状态成批刷新到输出物理点。

在实际工程中,有部分输入和输出需要优先快速处理时,可以使用 MOV_BIR 立刻输入和 MOV_BIW 立刻输出指令。但对于扩展单元的输出/输入使用 MOV_BIR 或 MOV_BIW 是没有立即输入或立即输出效果的。

在图 6-28 所示的程序中，M0.0 接通执行 MOV_BIR 指令时，IB0（I0.0 ~ I0.7）的物理点输入信号立刻传送到 VB10，而 I0.0 ~ I0.7 的映像区状态不更新。

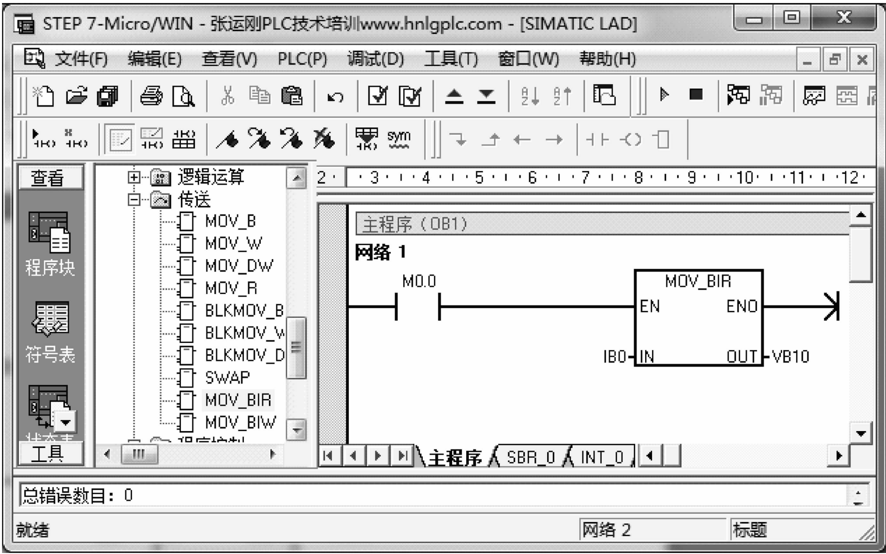


图 6-28　MOV_ BIR 指令应用

在图 6-29 所示的程序中，M0.0 接通在执行 MOV_BIW 指令时，VB10（V10.0 ~ V10.7）的状态立刻输出到输出物理点，而 QB0 的输出映像区还未更新。

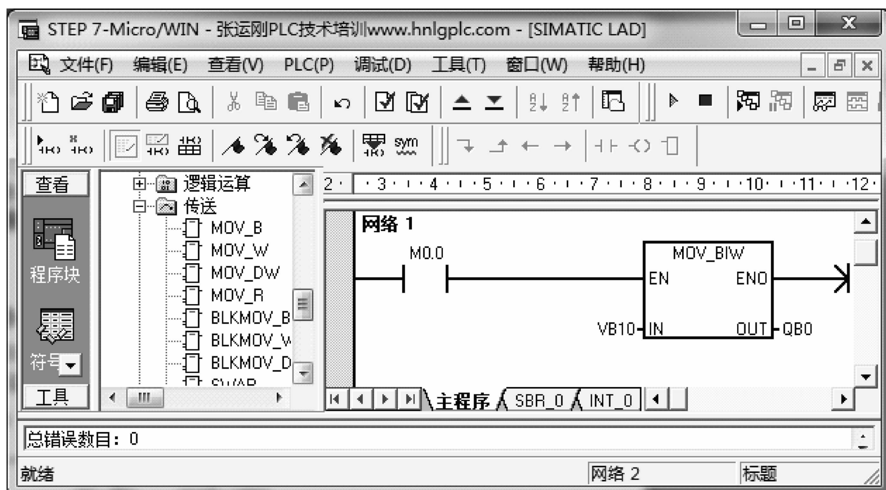


图 6-29 MOV_ BIW 指令应用

在图6-30 所示的程序中，当 I0. 0 和 I1. 0 同时接通，可以看到 Q0. 0 比 Q1. 0 先接通；当 I0. 0 和 I1. 0 同时断开，可以看到 Q0. 0 比 Q1. 0 先断开。本例请使用 S7-200 CPU224 或 CPU226。

在应用 MOV_BIR 和 MOV_BIW 指令时，下面两种情况需要注意的，一是脉冲执行与连续执行有区别，二是当使用间接寻址时需注意不要超出寻址范围。

在实现立即输入和立即输出时，还可以使用位指令。立即输入和立即输出指令的位指令包括如下 5 种：

-‖ -|I|-;

-‖ |/I|-;

-() -(I);

-() -(S I);

-() -(R I)。

普通输入和普通输出也称为扫描输入和扫描输出，也就是按照 PLC 的扫描周期来处理输入与输出。西门子 S7-200 PLC 对处理用户程序每个扫描周期分 3 个阶段执行：在扫描周期的开始进行输入刷新阶段，即读取物理点的状态保存在输入映像区，然后进入第二阶段运算用户程序，从上到下、从左到右逐行逐条进行运算，输入信号从输入映像区读取，运算输出的结果暂存输出映像区，最后是输出刷新阶段，即把输出映像区的状态成批传送到输出锁存器中。

在同一时间 PLC 只进行一个阶段的工作，在输入刷新阶段只将物理输入信号传送到输入映像区，不进行程序运算和输出刷新；在程序运算阶段只进行逐行逐条运算，不进行输入刷新和输出刷新工作；在输出刷新阶段只是把输出映像区的状态传送到输出锁存器，不进行输入刷新和程序运算。

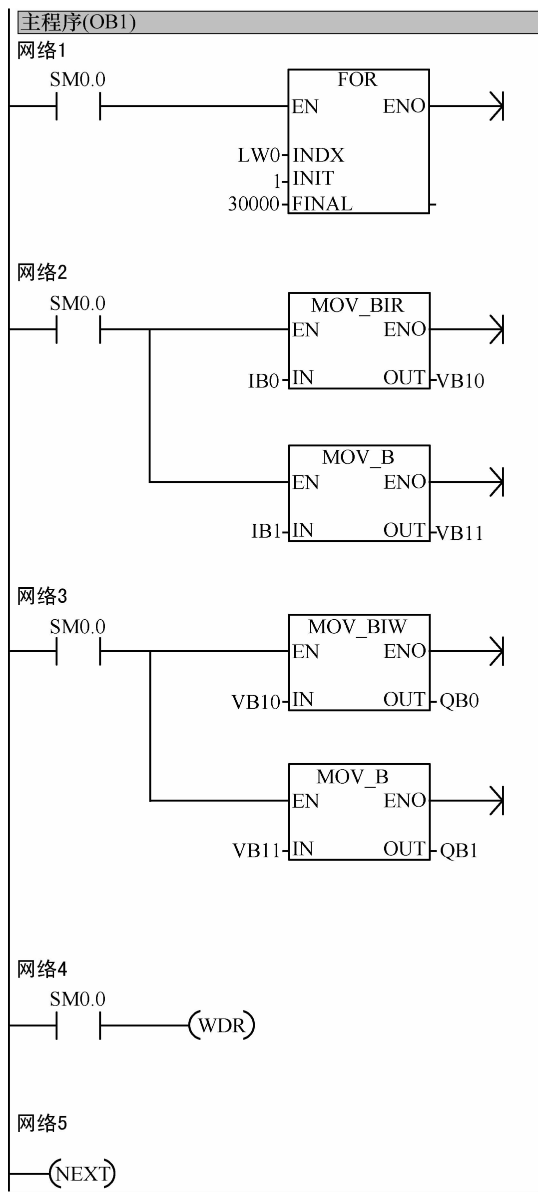


图 6-30　立即指令与普通指令输入/输出效果对比

如图 6-31 所示程序，在每个扫描周期的开始对物理点信号进行刷新，其中 I0. 0 和 I0. 1 物理点的状态分别传送到其对应的输入映像区；在第二个阶段运算程序阶段，先运算网络 1 的程序读取映像区，I0. 0 的常开触点状态驱动 Q0. 0，把运算结果 Q0. 0 的状态暂存输出映像区，再运算网络 2、网络 3 的程序；在最后阶段输出刷新阶段，把输出映像区的状态成批传送到输出锁存器。

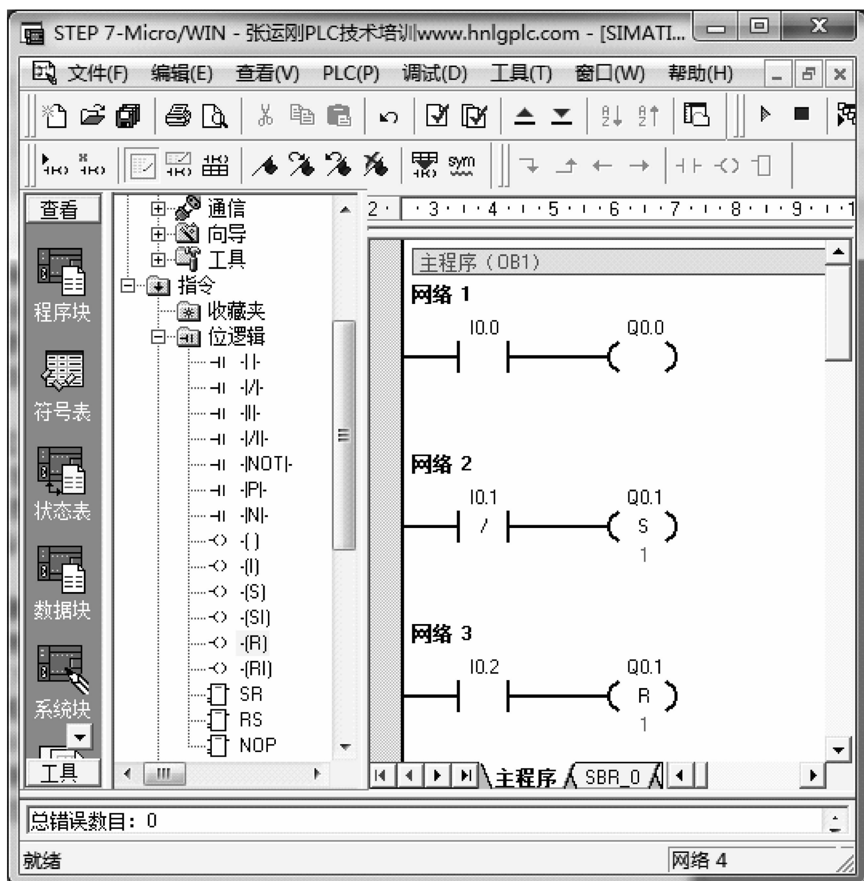


图 6-31 普通输入和普通输出程序

立即输入和立即输出不是按照扫描周期处理的。

在同一时间 PLC 原则上只进行一个阶段的工作，在输入刷新阶段只将物理输入信号传送到输入映像区，不进行程序运算和输出刷新；在程序运算阶段只进行逐行逐条运算，不进行输入刷新和输出刷新工作，但对于立即输入和立即输出的情况则不同，在运算程序的同时也对物理点信号读取，对于立即输出的则马上输出锁存器；在输出刷新阶段只是把输出映像区的状态传送到输出锁存器，不进行输入刷新和程序运算。

如图 6-32 所示程序，在每个扫描周期的开始对物理点信号进行刷新，其中 I0.0 和 I0.1 物理点的状态分别传送到其对应的输入映像区；在第二个阶段运算程序阶段，先运算网络 1 的程序，读取物理点的状态驱动 Q0.0，把运算结果 Q0.0 的状态传送输出锁存器的状态，再运算网络 2、网络 3 的程序，把运算结果输出到锁存器。

问 4：在应用 MOV_BIR 和 MOV_BIW 指令时需要注意些什么？

答：MOV_BIR 指令的源只能是有物理点的 I 输入，MOV_BIW 指令的目标只能是有物理点的 Q 输出。

在应用 MOV_BIR 和 MOV_BIW 指令时，下面两种情况需要注意的，一是脉冲执行与连续执行有区别，二是当使用间接寻址时需注意不要超出寻址范围。

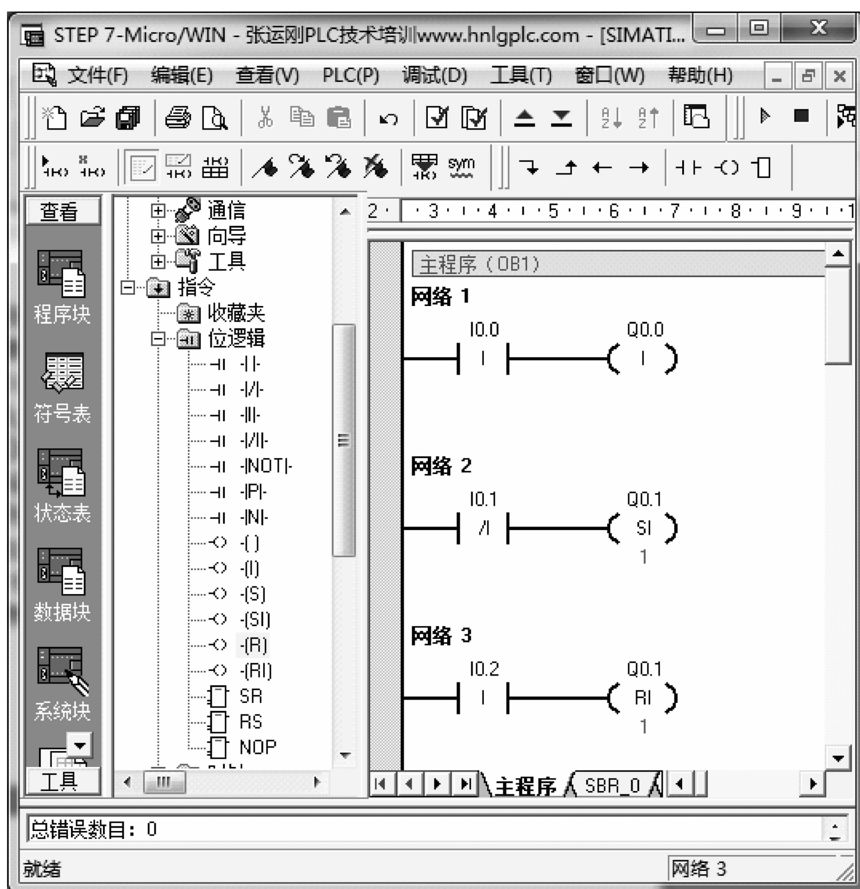


图 6-32　立即输入和立即输出程序

第 7 章　触点比较指令

7.1　数值比较 =/≠/>/</≥/≤

> 问：数值比较指令基本功能是什么？
> 　　数值比较指令样式是怎样的？
> 　　如何应用数值比较指令？
> 　　在应用数值比较指令时需要注意些什么？

问 1：数值比较指令基本功能是什么？

答：数值比较指令，其实是触点比较指令，当两个源的数值满足比较条件时那么这个触点就表示接通的，如果不满足比较条件就表示不通。

数值比较指令按照两个源的数值类型不同，分字节、字、双字和小数；按照比较方式不同，分 =（等于）、≠（不等于）、>（大于）、<（小于）、≥（大于等于）和≤（小于等于）。

在字节、字、双字和小数比较指令中，操作数可能的范围如图 7-1 所示。

字节比较 输入/输出 输入	操作数 IB,QB,MB,SMB,VB,SB,LB,AC, 常量,*VD,*LD,*AC	数据类型 字节
整数比较 输入/输出 输入	操作数 IW,QW,MW,SW,SMW,T,C,VW, LW,AIW,AC,常量,*VD,*LD,*AC	数据类型 整数
双整数比较 输入/输出 输入	操作数 ID,QD,MD,SD,SMD,VD,LD,HC, AC,常量,*VD,*LD,*AC	数据类型 双整数
小数比较 输入/输出 输入	操作数 ID,QD,MD,SD,SMD,VD,LD,HC, AC,常量,*VD,*LD,*AC	数据类型 小数

图 7-1　比较指令操作数的范围

问 2：数值比较指令样式是怎样的？

答：数值比较指令的样式见表 7-1。

表 7-1　数值比较指令样式分类表

<table>
<tr><th></th><th>=（等于）</th><th>≠（不等于）</th><th>>（大于）</th><th><（小于）</th><th>≥（大于等于）</th><th>≤（小于等于）</th></tr>
<tr><td>字节</td><td>-|==B|-</td><td>-|<>B|-</td><td>-|>B|-</td><td>-|<B|-</td><td>-|>=B|-</td><td>-|<=B|-</td></tr>
<tr><td>字</td><td>-|==I|-</td><td>-|<>I|-</td><td>-|>I|-</td><td>-|<I|-</td><td>-|>=I|-</td><td>-|<=I|-</td></tr>
<tr><td>双字</td><td>-|==D|-</td><td>-|<>D|-</td><td>-|>D|-</td><td>-|<D|-</td><td>-|>=D|-</td><td>-|<=D|-</td></tr>
<tr><td>小数</td><td>-|==R|-</td><td>-|<>R|-</td><td>-|>R|-</td><td>-|<R|-</td><td>-|>=R|-</td><td>-|<=R|-</td></tr>
</table>

问 3：如何应用数值比较指令？

答：字节、字、双字和小数比较指令应用程序如图 7-2 ~ 图 7-5 所示。

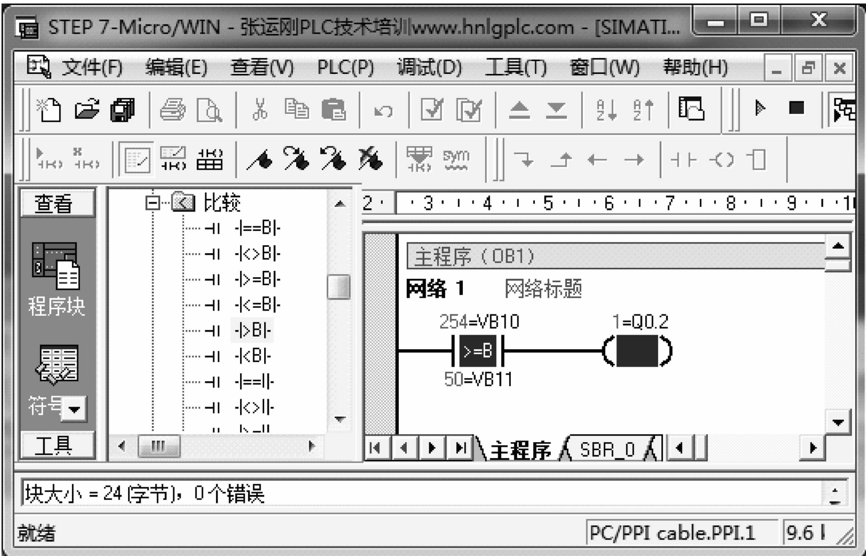


图 7-2　字节比较指令

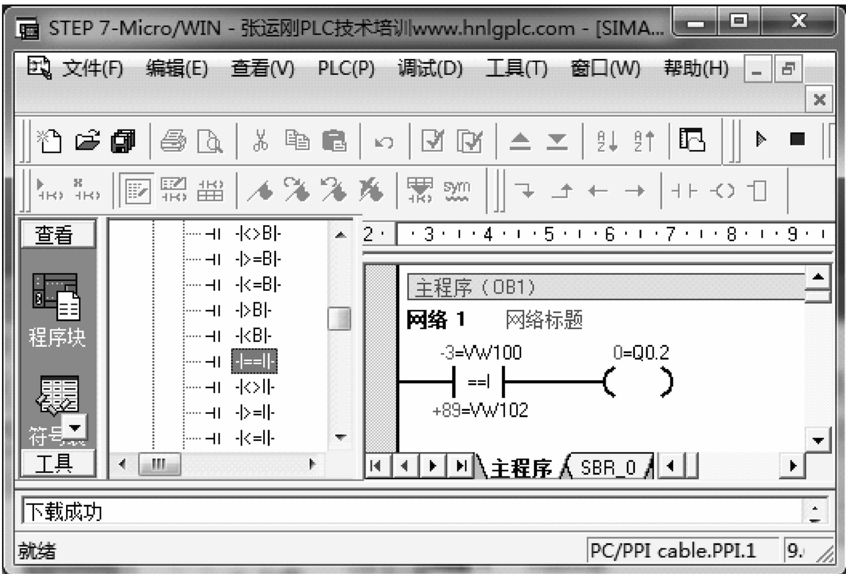


图 7-3　字比较指令

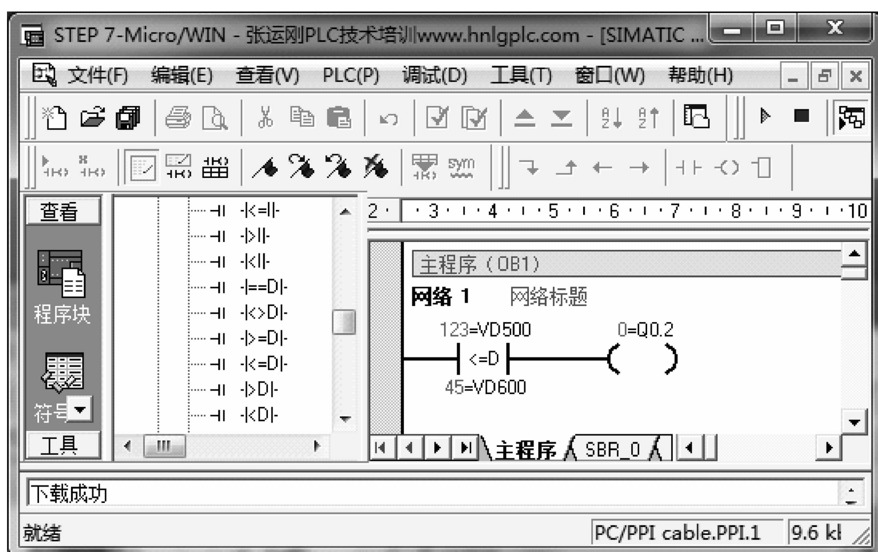


图 7-4　双字比较指令

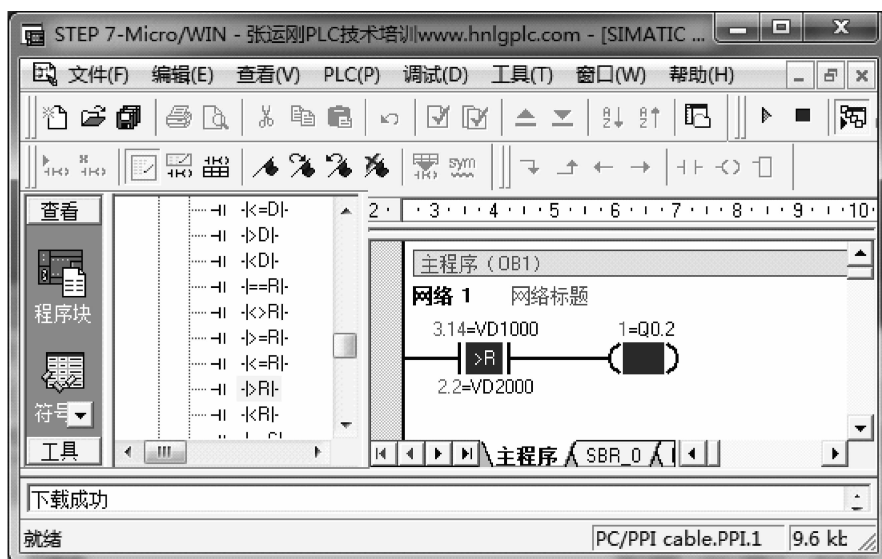


图 7-5　小数比较指令

如果要求完成如图 7-6 所示交通灯的控制，可以使用触点比较指令编写。实现程序如图 7-7 或者图 7-8 或者图 7-9 所示。

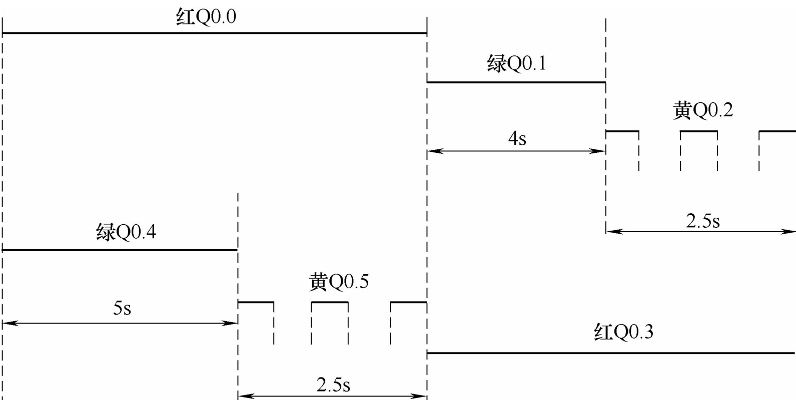


图 7-6　交通灯控制时序图

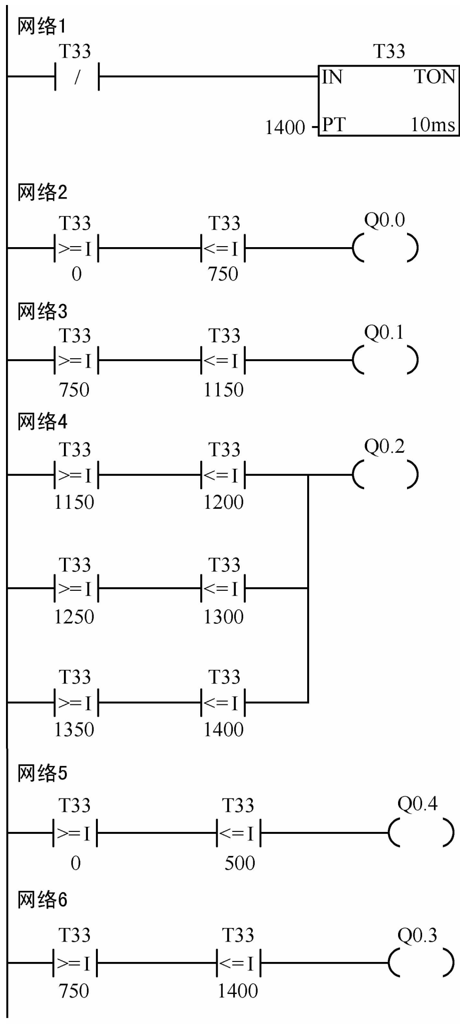


图 7-7　交通灯控制程序 1

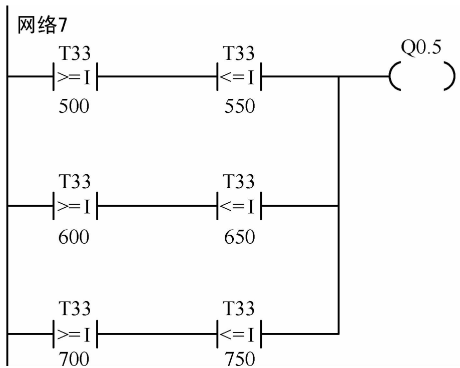


图 7-7　交通灯控制程序 1（续）

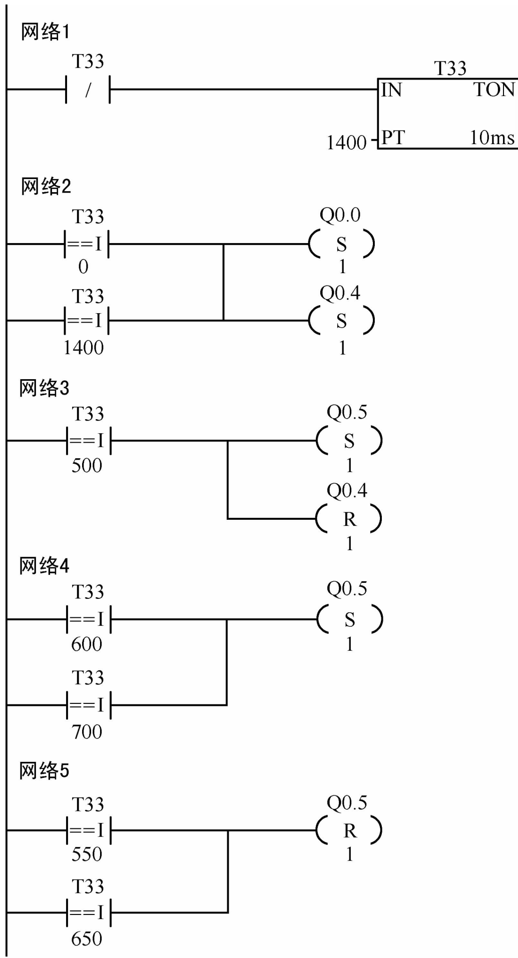


图 7-8　交通灯控制程序 2

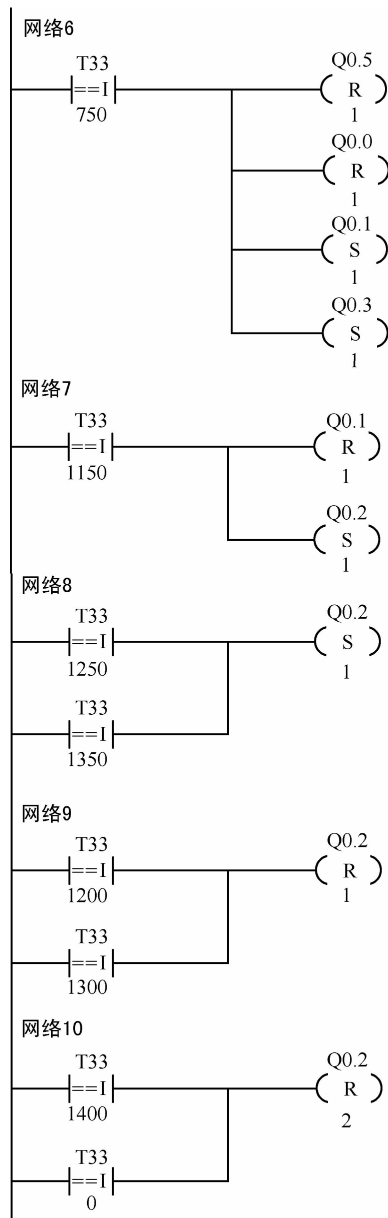


图 7-8　交通灯控制程序 2（续）

问 4：在应用数值比较指令时需要注意些什么？

答：使用数值比较指令时，需要注意以下两种情况：

一是，字节比较是两个无符号数进行比较，16 位整数、32 位整数和小数比较是两个有符号数进行比较。当双字整数比较时确保两个源数据都是双整数，同样当小数比较时确保两个源数据都是小数。

二是，当使用间接给定软元件，注意软元件超出范围的错误。

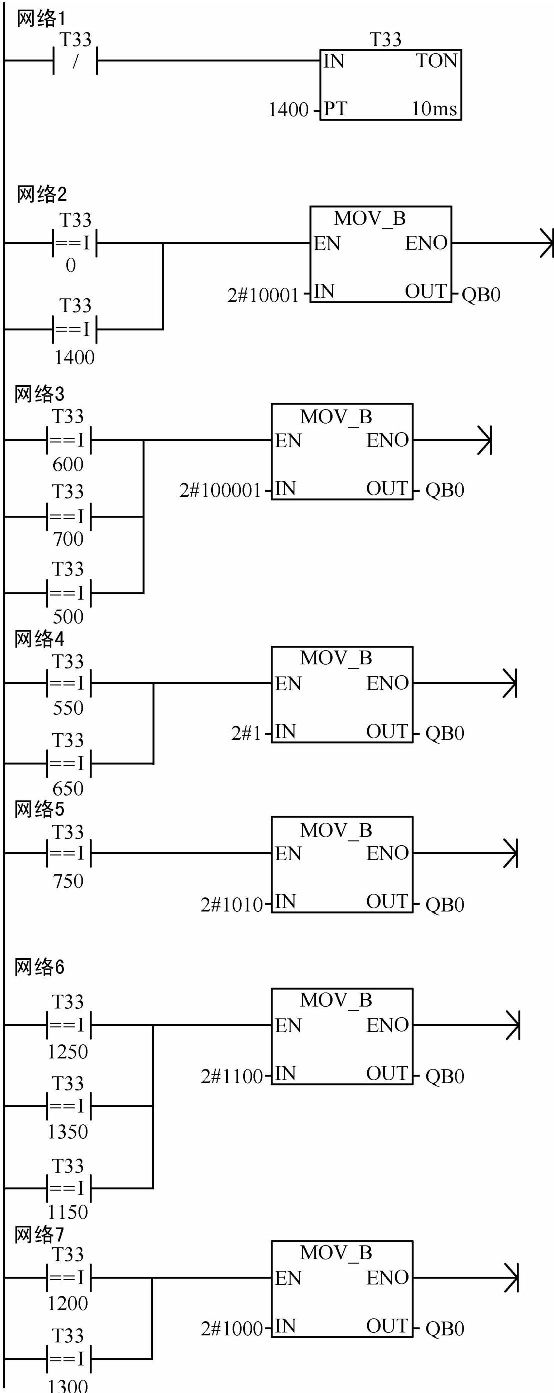


图 7-9　交通灯控制程序 3

7.2　字符串比较 =/≠

> 问：字符串比较指令基本功能是什么？
> 　　字符串比较指令样式是怎样的？
> 　　如何应用字符串比较指令？
> 　　在应用字符串比较指令时需要注意些什么？

问 1：字符串比较指令基本功能是什么？

答：执行字符串比较指令时，对两个源的字符进行比较，如果条件满足，表示该触点是接通的，如果条件不满足，则该触点是断开的。

ASCII 常数字符串数据类型的格式是：一系列字符和对应的内存地址，每个字符作为一个字节存储。字符串的第一个字节是定义字符串长度（即字符数）的整数。如果常数字符串被直接输入程序编辑器或数据块，那么该字符串必须用双引号字符起始和结束（“字符串常数”）。

字符串数据类型的格式见表 7-2 所示。字符串的长度可以是 0 ~ 254 字符。字符串的最大长度是 255 个字节（长度字节加上 254 个字符的字节）。

表 7-2　字符串数据类型的格式

字节 0	字节 1	字节 2	…	字节 254
长度	字符 1	字符 2	…	字符 254

字符串比较指令中，由于字符串不是数值所以字符串没有大小之分，只有相同和不同的比较。

问 2：字符串比较指令样式是怎样的？

答：字符串比较指令的样式如图 7-10 所示。

数入/输出	操作数	数据类型
┤==S├ IN1	VB,常数字符串,LB,*VD*LD, *AC	字符串
┤<>S├ IN2	VB,LB,*VD,*LD,*AC	字符串

图 7-10　字符串比较指令样式

问 3：如何应用字符串比较指令？

答：在工业应用领域，启动信号很多时候来自扫描枪扫描回来的信息，使用字符串比较指令满足条件就控制输出；同样，停止信号很多时候也来自扫描枪扫描回来的信息，使用字符串比较指令满足条件就控制停止输出。这样就可以实现更加智能的“会认字”自动控制。

图 7-11 的所示应用程序在 RUN 模式下，当按动 I0.2 按钮，这时看到 Q0.0 有输出，称为启动；当按动 I0.3 按钮，这时看到 Q0.0 没有输出了，称为停止。

问 4：在应用字符串比较指令时需要注意些什么？

答：在使用字符串比较指令时需要注意以下几种情况：

一是遇到非法间接地址。

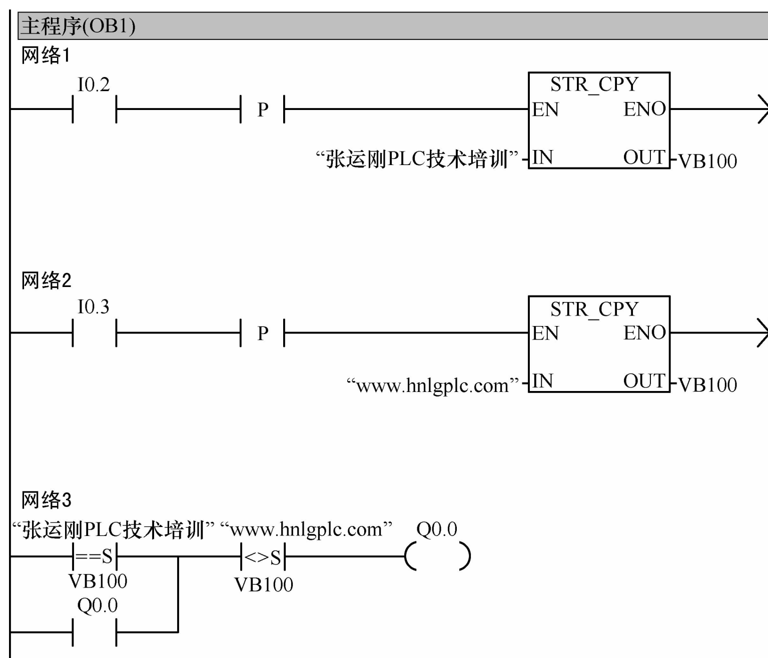


图 7-11　字符串指令应用

二是遇到长度超过 254 个字符的字符串。

三是字符串的起始地址和长度无法放入一个指定的连续内存区。为了防止出现这类错误，在执行字符串比较指令之前，需要以合适的方式初始化指针和用于放置 ASCII 字符串的内存位置，并确认 ASCII 字符串保留的缓冲区可完全放置在指定的内存区中。

第 8 章　数学运算和转换指令

8.1　整数运算

问：整数的特征是什么？ 整数运算指令样式是怎样的？ 整数运算指令基本运算规律怎样？ 在应用整数运算指令时需要注意些什么？

问 1：整数的特征是什么？

答：在 S7-200 CPU 中，整数分 16 位整数和 32 位整数。不管是 16 位还是 32 位整数，其最高一位都是符号位，符号位为“0”是正数，符号位为“1”是负数。正数使用原码存放，负数使用补码存放。16 位的原码和补码见表 8-1，表中“ +5”是原码，“ -5”是补码。

表 8-1　16 位原码和补码

16 位十进制数	符号	数值														
	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
+5	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1
-5	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1

问 2：整数运算指令样式是怎样的？

答：整数四则运算指令包括“ +”、“ -”、“ ×”、“ ÷”的 16 位整数和 32 位整数等的四则运算指令。这些指令样式如图 8-1 ~ 图 8-10 所示。

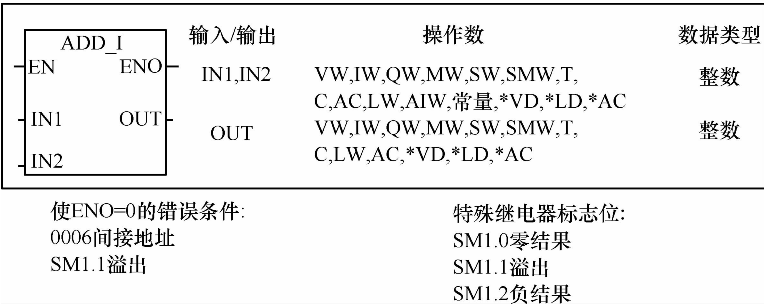


图 8-1　16 位加法

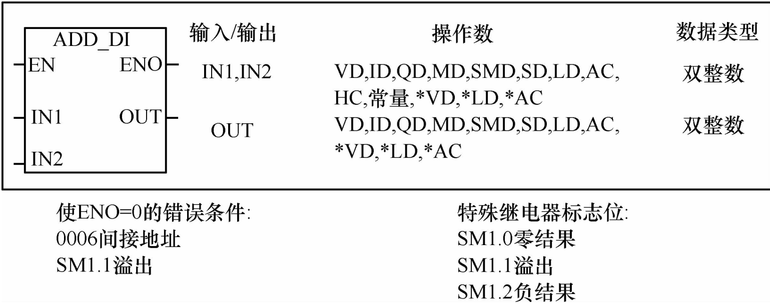


图 8-2　32 位加法

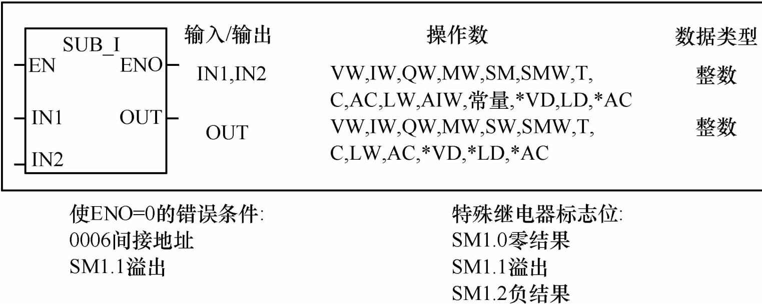


图 8-3　16 位减法

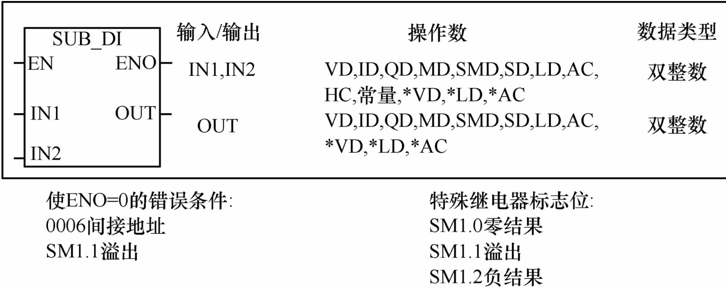


图 8-4　32 位减法



图 8-5　16 位乘法

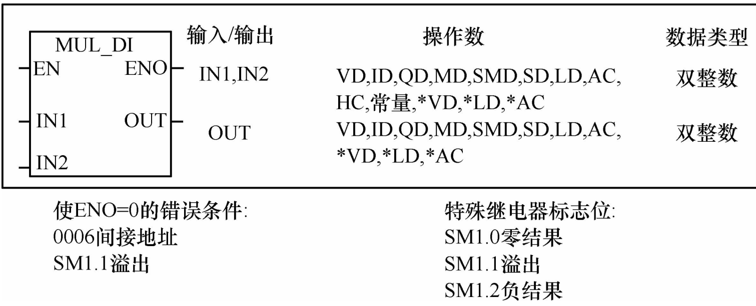


图 8-6　32 位乘法



图 8-7　32 位乘积的 16 位乘法

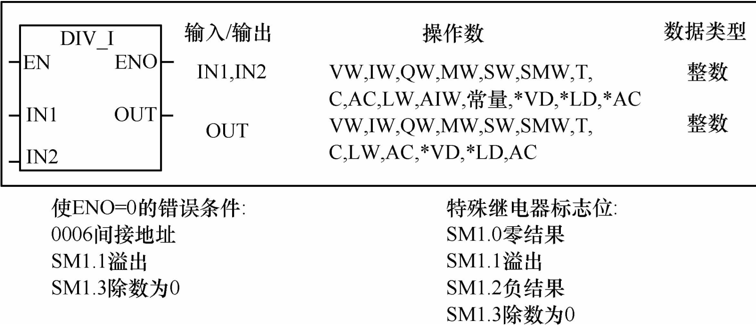


图 8-8　16 位除法

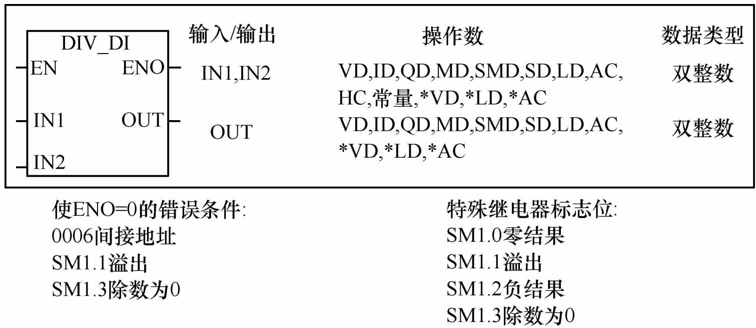


图 8-9　32 位除法

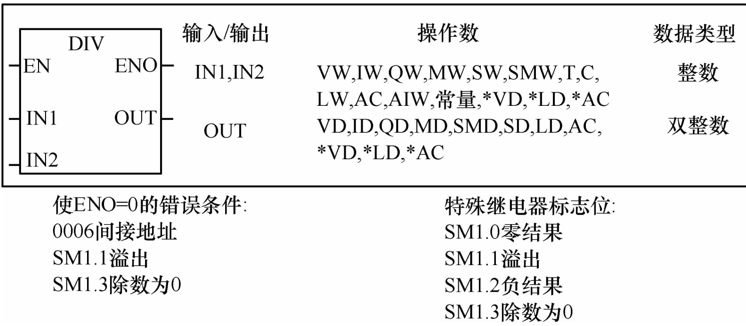


图 8-10　带余数的 16 位除法

问 3：整数运算指令基本运算规律怎样?

答：在执行加法、减法、乘法和除法等整数数学运算指令时，如果 SM1.1 溢出标志位等于“1”，表示结果有溢出，说明结果是错误的；如果 SM1.2 负标志位等于“1”，表示结果是负数；如果 SM1.0 为“1”说明结果为零；在除法整数运算时，当除数为“0”，将不执行该除法指令，同时特殊继电器标志位 SM1.3 为“1”。

在执行 16 位整数加法指令 ADD_I 时，IN1 + IN2 = OUT，同时特殊继电器标志位会动作。

在执行图 8-11 所示的 ADD_I 指令时，当 M0.1 接通，IN1 源 1 VW10 的数值加上 IN2 源 2 VW12 的数值把结果传送到 OUT 目标 VW14（即是 VW10 + VW12 = VW14），各个数据都是有符号数，二进制中最高位是符号位，“0”表示正数，“1”表示负数。

当加的结果为 0，零标志位 SM1.0 会变为“1”。

当加的结果大于 32767 或小于 -32768，溢出标志位 SM1.1 会变为“1”。

当加的结果在 -1 ~ -32768，负结果标志位 SM1.2 会变为“1”。

在图 8-11 所示的程序中，调试的结果见表 8-2 所示。

表 8-2　ADD_I 调试结果

被加数 VW10	加数 VW12	和数 VW14	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	ENO Q0.3
0	0	0	1	0	0	1
2000	30201	32201	0	0	0	1
-330	-542	-872	0	0	1	1
20000	32000	-13536	0	1	1	0
-5	-32768	32763	0	1	0	0
-1	-32768	32767	0	1	0	0
1	32767	-32768	0	1	1	0

在图 8-12 所示的程序中,其中一个源和目标的软元件地址相同(都是 VW10),如果使用连续执行型指令要注意,因为在 M0.1 接通时,每个扫描周期都在执行加法指令。这种情况考虑使用脉冲执行型指令,如图 8-13 所示程序。

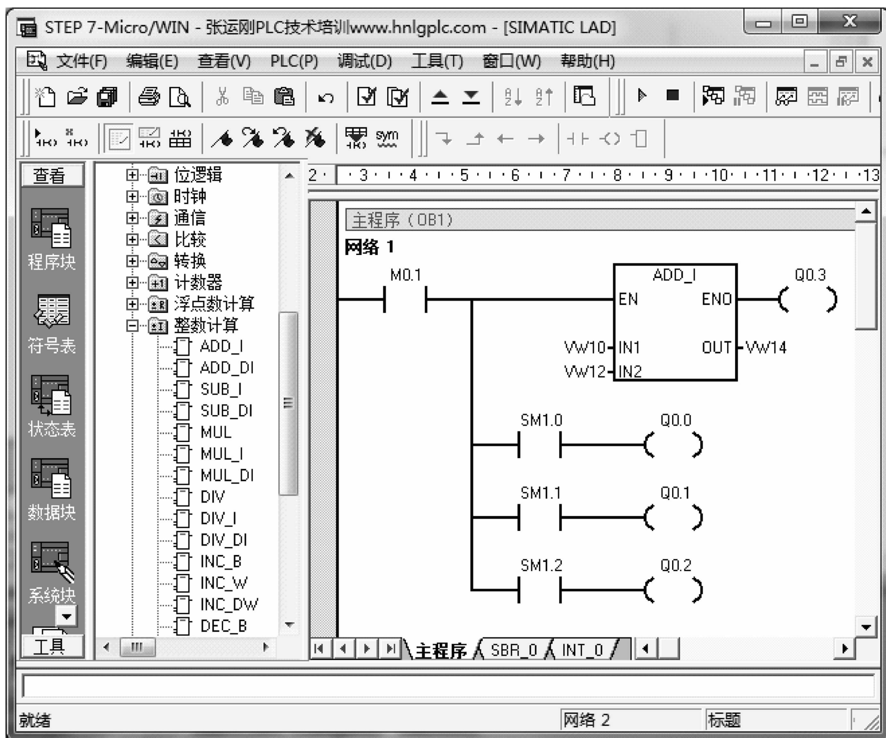


图 8-11　ADD_I 指令应用 1

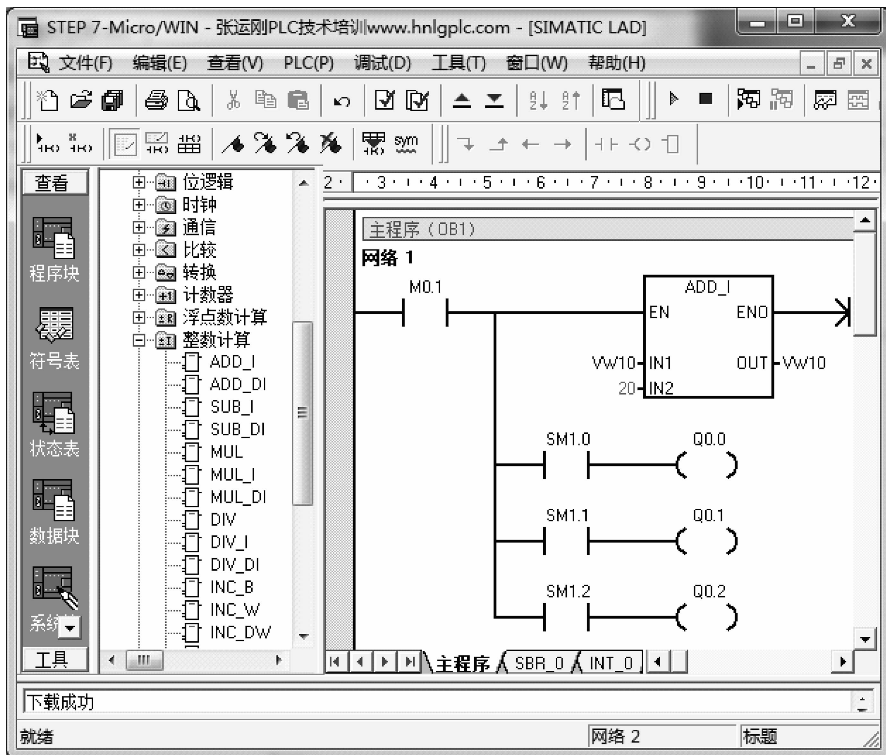


图 8-12　ADD_ I 指令应用 2

ADD_DI32 位整数运算指令的规律类似于 ADD_I16 位整数运算指令，不同的是 ADD_DI 的操作数是 32 位整数，而 ADD_I 的操作数是 16 位整数。

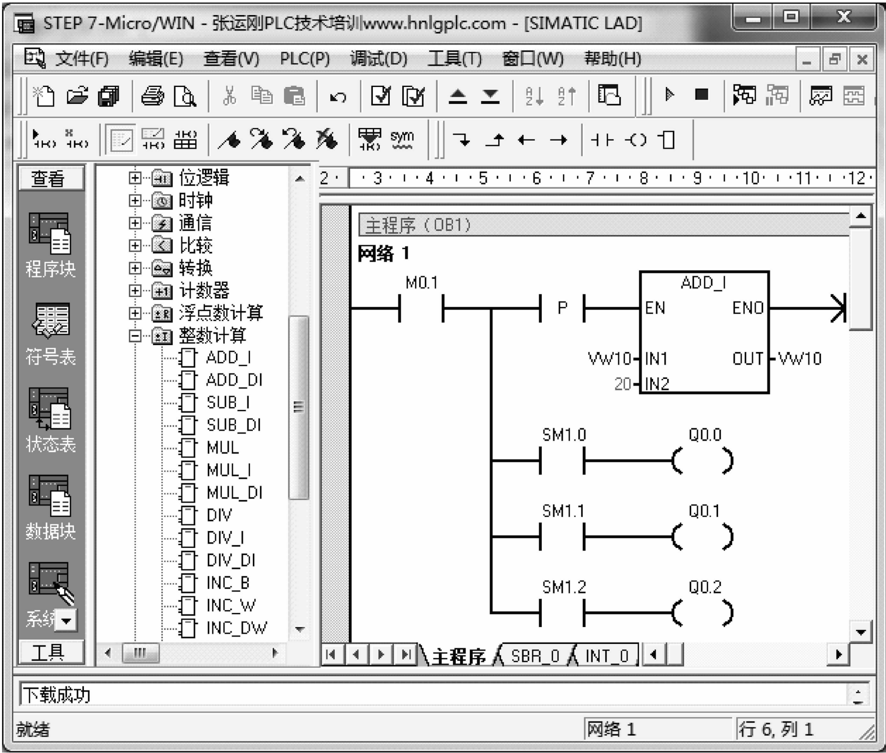


图 8-13　ADD_I 指令应用 3

在执行 16 位整数减法指令 SUB_I 时，IN1 + IN2 = OUT，同时特殊继电器标志位会动作。

在图 8-14 所示的 SUB_I 指令时，当 I0.1 接通，IN1 源 1 VW10 的数值减上 IN2 源 2 VW12 的数值把结果传送到 OUT 目标 VW14（即是 VW10 − VW12 = VW14），各个数据都是有符号数，二进制中最高位是符号位，“0”表示正数，“1”表示负数。

当减的结果为 0，零标志位 SM1.0 会变为“1”。

当减的结果大于 32767 或小于 −32768，溢出标志位 SM1.1 会变为“1”。

当减的结果在 −1 ~ −32768，负结果标志位 SM1.2 会变为“1”。

在图 8-14 所示的程序中，调试的结果见表 8-3。

表 8-3　SUB_I 调试结果

被减数 VW10	减数 VW12	和数 VW14	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	ENO Q0.3
0	0	0	1	0	0	1
2000	−30201	32201	0	0	0	1
−330	542	−872	0	0	1	1
20000	−32000	−13536	0	1	1	0
−5	32768	32763	0	1	0	0
−1	32768	32767	0	1	0	0
1	−32767	−32768	0	1	1	0

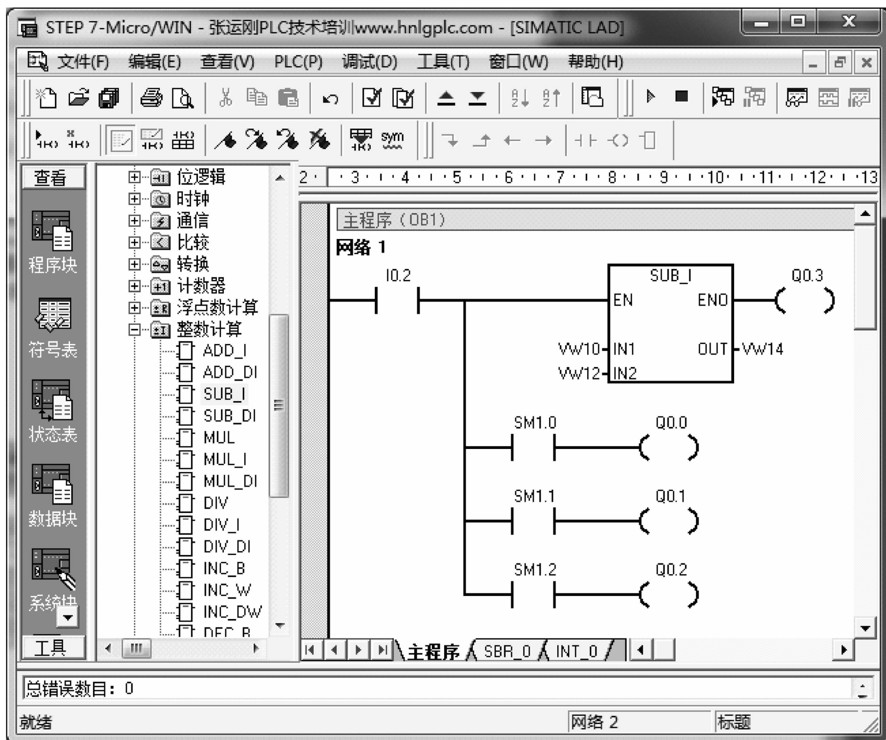


图 8-14 SUB_ I 指令应用 1

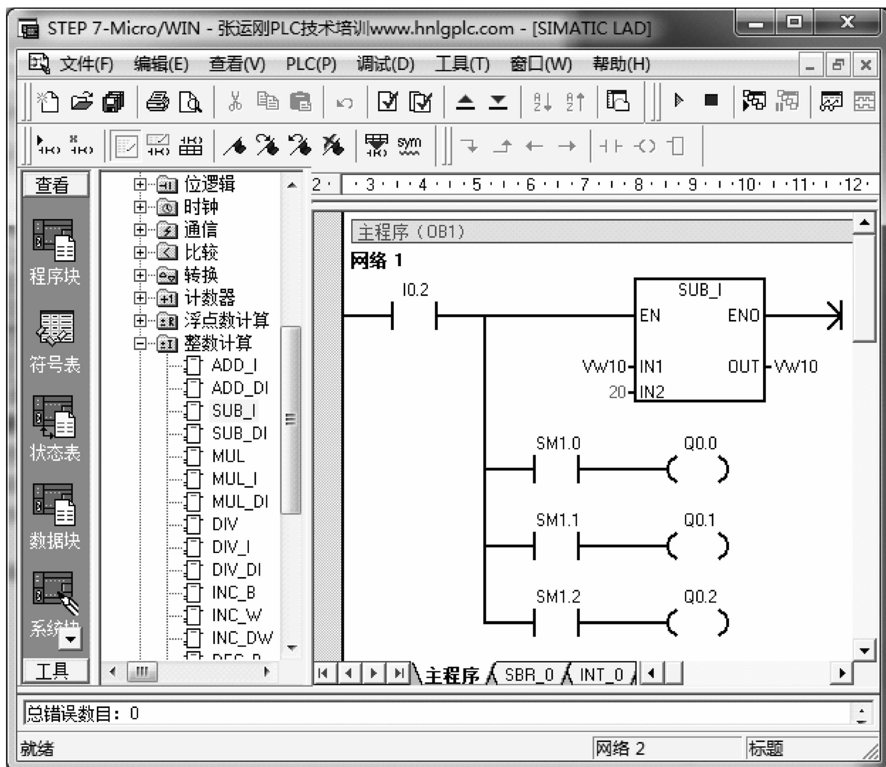


图 8-15 SUB_ I 指令应用 2

在图 8-15 所示的程序中，其中一个源和目标的软元件地址相同（都是 VW10），如果使用连续执行型指令要注意，因为在 I0.1 接通时，每个扫描周期都在执行减法指令。这种情况考虑使用脉冲执行型指令，如图 8-16 所示程序。

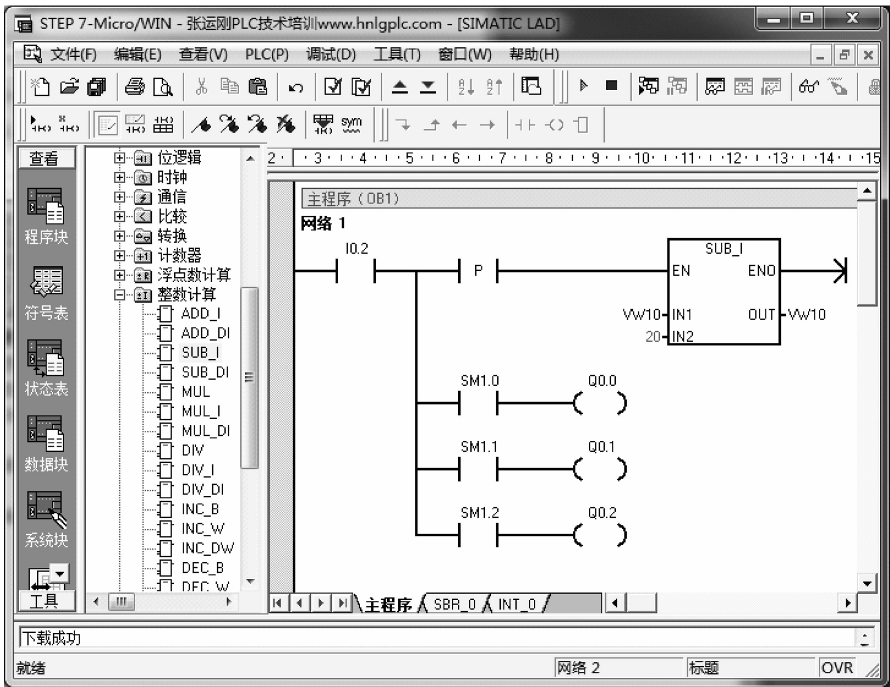


图 8-16　SUB_I 指令应用 3

SUB_DI32 位整数运算指令的规律类似于 SUB_I16 位整数运算指令，不同的是 SUB_DI 的操作数是 32 位整数，而 SUB_I 的操作数是 16 位整数。

在执行 16 位整数乘法指令 MUL_ I 时，IN1 × IN2 = OUT（16 位 × 16 位 = 16 位），同时特殊继电器标志位会动作。

在图 8-17 所示程序中，当 I0.2 接通，IN1 源 1 VW10 的数值乘以 IN2 源 2 VW12 的数值，把结果传送到 OUT 目标 VW14（即是 VW10 × VW12 = VW14），各个数据都是有符号数，二进制中最高位是符号位，“0” 表示正数，“1” 表示负数。

当乘的结果为 0，零标志位 SM1.0 会变为“1”。

当乘的结果大于 32767 或小于 -32768，溢出标志位 SM1.1 会变为“1”。

当乘的结果在 -1 ~ -32768，负结果标志位 SM1.2 会变为“1”。

图 8-17 的程序调试结果见表 8-4。

表 8-4　MUL_I 调试结果

被乘数 VW10	乘数 VW12	积数 VW14	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	ENO Q0.3
100	0	0	1	0	0	1
-2	15000	-30000	0	0	1	1
2	15000	30000	0	0	0	1

（续）

被乘数 VW10	乘数 VW12	积数 VW14	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	ENO Q0.3
2	-32000	2	0	1	0	0
-5	32767	-5	0	1	0	0
2	32767	2	0	1	0	0
2	-32768	2	0	1	0	0

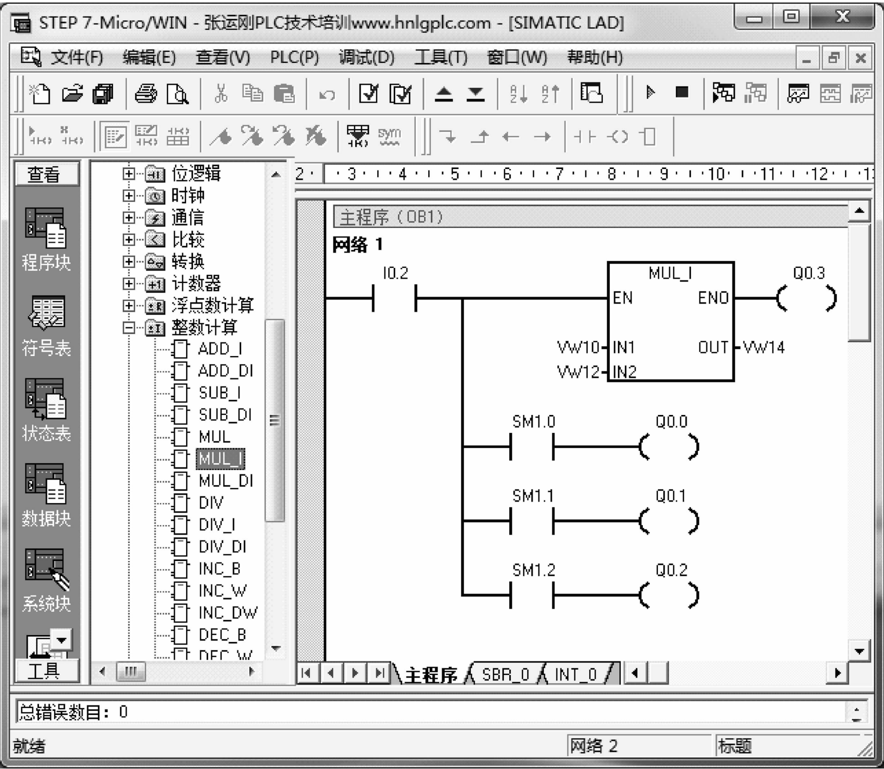


图 8-17　MUL_I 指令应用 1

在图 8-18 所示的程序中，其中一个源和目标的软元件地址相同（都是 VW10），如果使用连续执行型指令要注意，因为在 I0.2 接通时，每个扫描周期都在执行乘法指令。这种情况考虑使用脉冲执行型指令，如图 8-19 所示。

MUL_DI32 位整数运算指令的规律类似于 MUL_I16 位整数运算指令，不同的是 MUL_DI 的操作数是 32 位整数，而 MUL_I 的操作数是 16 位整数。

在执行 16 位整数除法指令 DIV_ I 时，IN1 ÷ IN2 = OUT（16 位整数 ÷ 16 位整数 = 16 位整数），不保留余数，同时特殊继电器标志位会动作。

在图 8-20 所示程序中，当 I0.2 接通，IN1 源 1 VW10 的数值除以 IN2 源 2 VW12 的数值把结果传送到 OUT 目标 VW14（VW10 ÷ VW12 = VW14），各个数据都是有符号数，二进制中最高位是符号位，“0”表示正数，“1”表示负数。

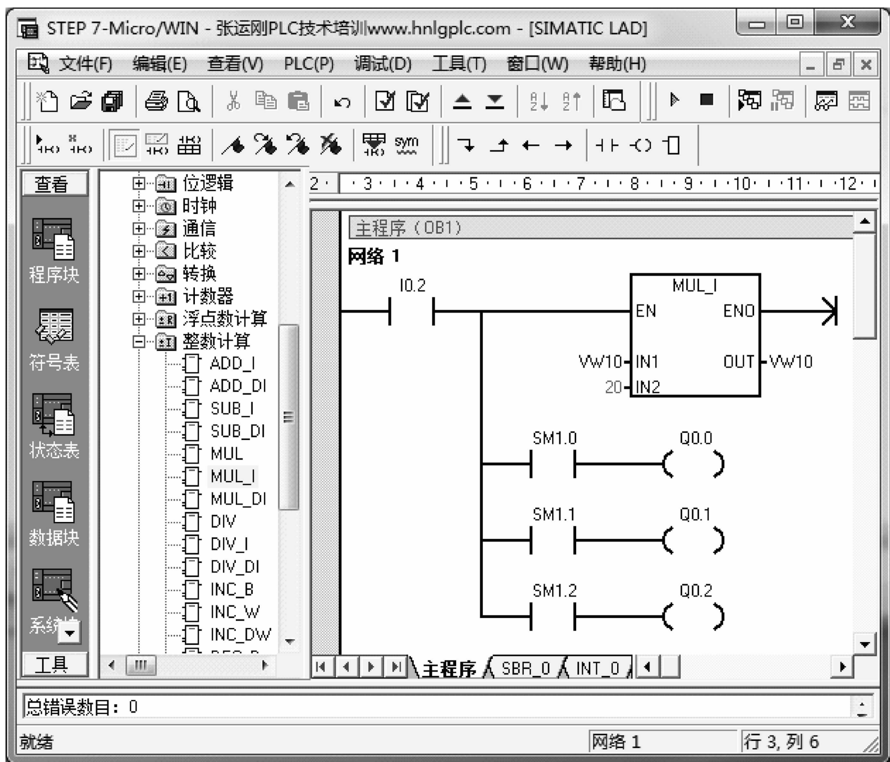


图 8-18　MUL_ I 指令应用 2

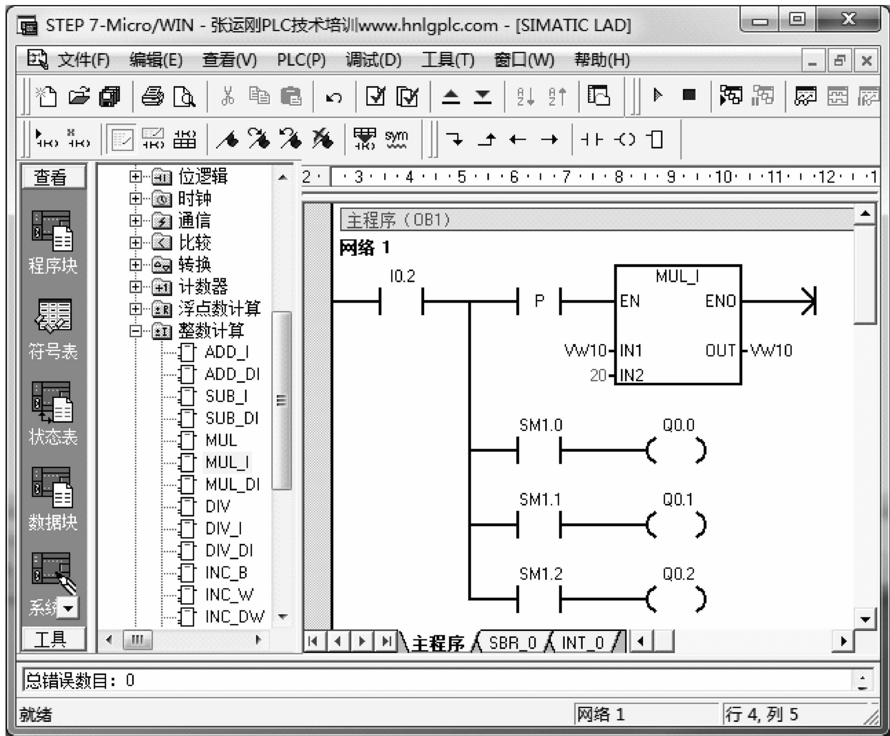


图 8-19　MUL_ I 指令应用 3

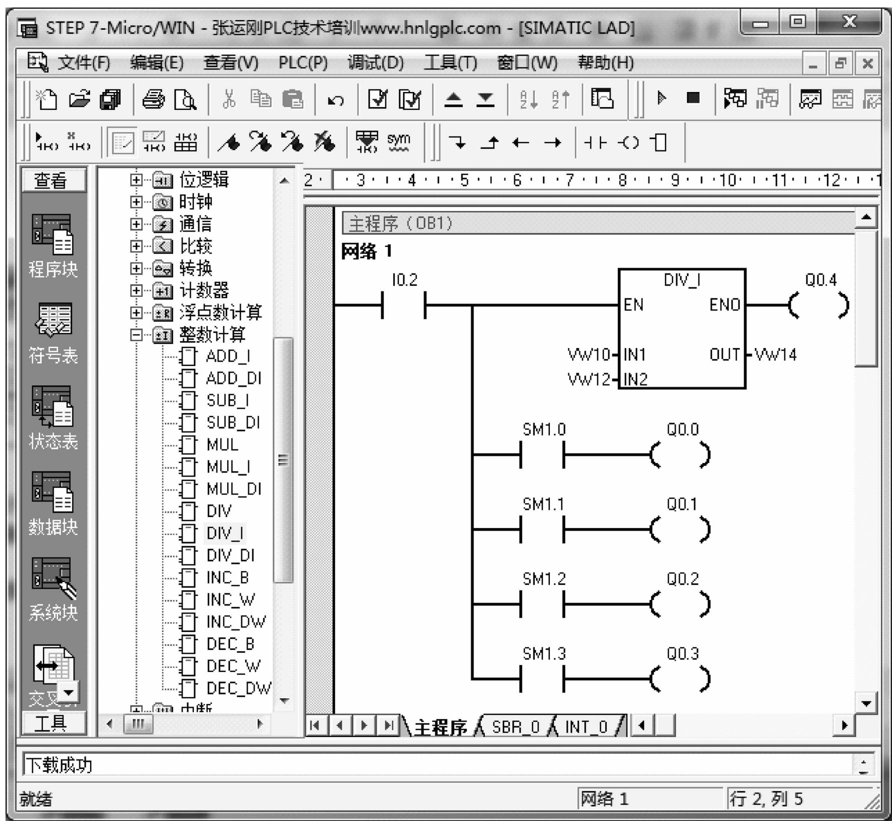


图 8-20　DIV_I 指令应用 1

当除的结果为 0，零标志位 SM1. 0 会变为“1”。

当除的结果大于 32767 或小于 -32768，溢出标志位 SM1. 1 会变为“1”。

当除的结果在 -1 ~ -32768，负结果标志位 SM1. 2 会变为“1”。

当除数为零，除数为零标志位 SM1. 3 会变为“1”。

图 8-20 所示的程序调试结果见表 8-5。

表 8-5　DIV_I 调试结果

被除数 VW10	除数 VW12	商数 VW14	零标志位 SM1. 0(Q0. 0)	溢出标志位 SM1. 1(Q0. 1)	负标志位 SM1. 2(Q0. 2)	除数为零标志位 SM1. 3(Q0. 3)	ENO Q0. 4
0	100	0	1	0	0	0	1
-300	2	-150	0	0	1	0	1
4200	2	2100	0	0	0	0	1
4200	0	4200	0	0	0	1	0
-5	32767	0	1	0	0	0	1

在图 8-21 所示的程序中，其中一个源和目标的软元件地址相同（都是 VW10），如果使用连续执行型指令要注意，因为在 I0. 2 接通时每个扫描周期都在执行除法指令。这种情况考虑使用脉冲执行型指令，如图 8-22 所示。

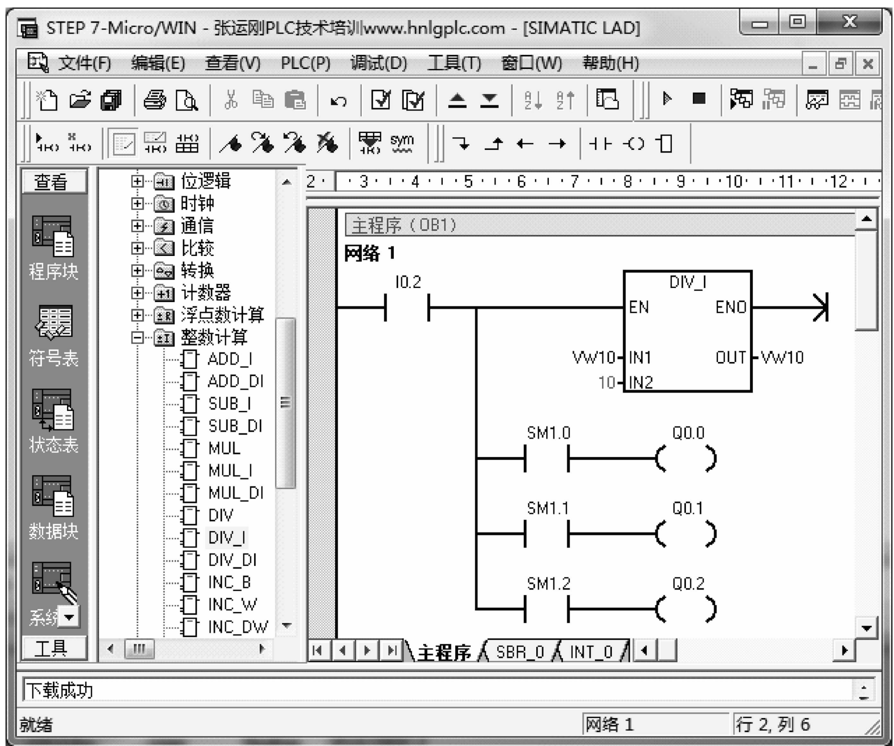


图 8-21　DIV_I 指令应用 2

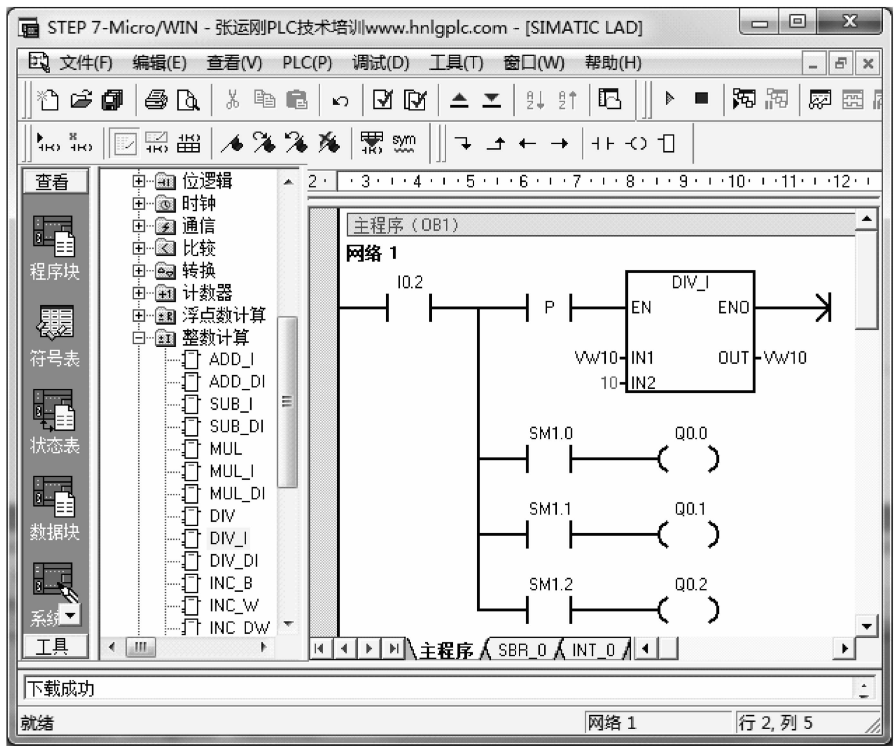


图 8-22　DIV_I 指令应用 3

DIV_DI32 位整数运算指令的规律类似于 DIV_I16 位整数运算指令，不同的是 DIV_DI 的操作数是 32 位整数，而 DIV_I 的操作数是 16 位整数。

在执行 MUL 乘法指令时，IN1 × IN2 = OUT（16 位整数 × 16 位整数 = 32 位整数），同时特殊继电器标志位会动作。

在图 8-23 所示的程序中，当 I0.2 接通，在执行 MUL 32 位乘法指令时，IN1 × IN2 = OUT（即是 VW10 × VW12 = VD14），同时特殊继电器标志位会动作。各个数值都是有符号数，二进制中最高位是符号位，“0”表示正数，“1”表示负数。比如 VW10 = 32767，VW12 = 300，当 I0.2 接通时，VD14 = 9830100。

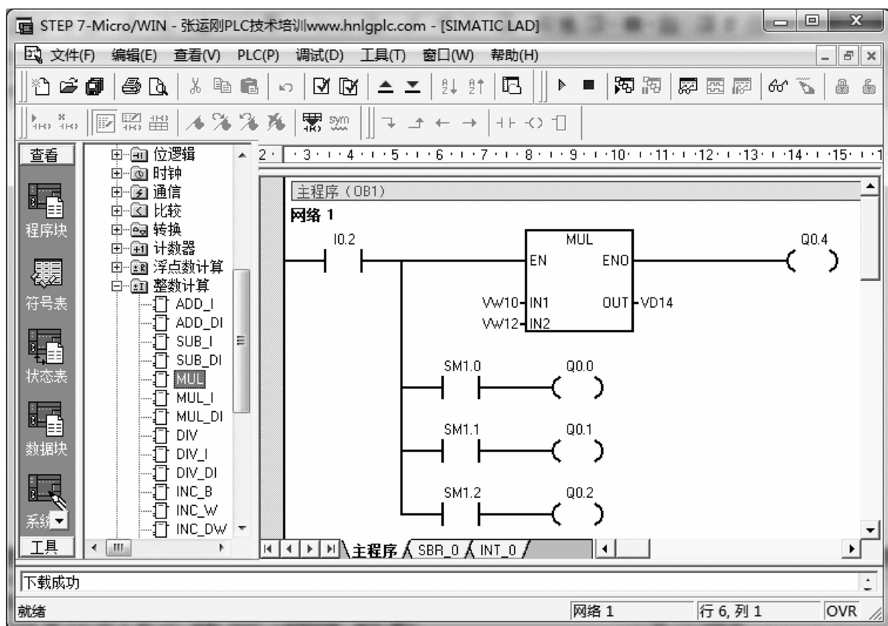


图 8-23　MUL 乘法指令应用

当乘的结果为 0，零标志位 SM1.0 会变为“1”。

当乘的结果大于 2147483647 或小于 -2147483648，溢出标志位 SM1.1 会变为“1”。

当乘的结果在 -1 ~ -2147483648，负结果标志位 SM1.2 会变为“1”。

在执行 DIV 除法指令时，IN1 ÷ IN2 = OUT（16 位整数 ÷ 16 位整数 = 32 位整数），保留余数同时特殊继电器标志位会动作。

在图 8-24 所示的程序中，当 I0.2 接通，在执行 DIV 32 位除法指令时，IN1 ÷ IN2 = OUT（即是 VW10 ÷ VW12 = VW16……VW14，VW16 是商数、VW14 是余数），同时特殊继电器标志位会动作。各个数值都是有符号数，二进制中最高位是符号位，“0”表示正数，“1”表示负数。比如 VW10 = 10，VW12 = 3，当 I0.2 接通时，VW16 = 3、VW14 = 1。

当除的结果为 0，零标志位 SM1.0 会变为“1”。

当除的结果大于 2147483647 或小于 -2147483648，溢出标志位 SM1.1 会变为“1”。

当除的结果在 -1 ~ -2147483648，负结果标志位 SM1.2 会变为“1”。

当除数为零，除数为零标志位 SM1.3 会变为“1”。

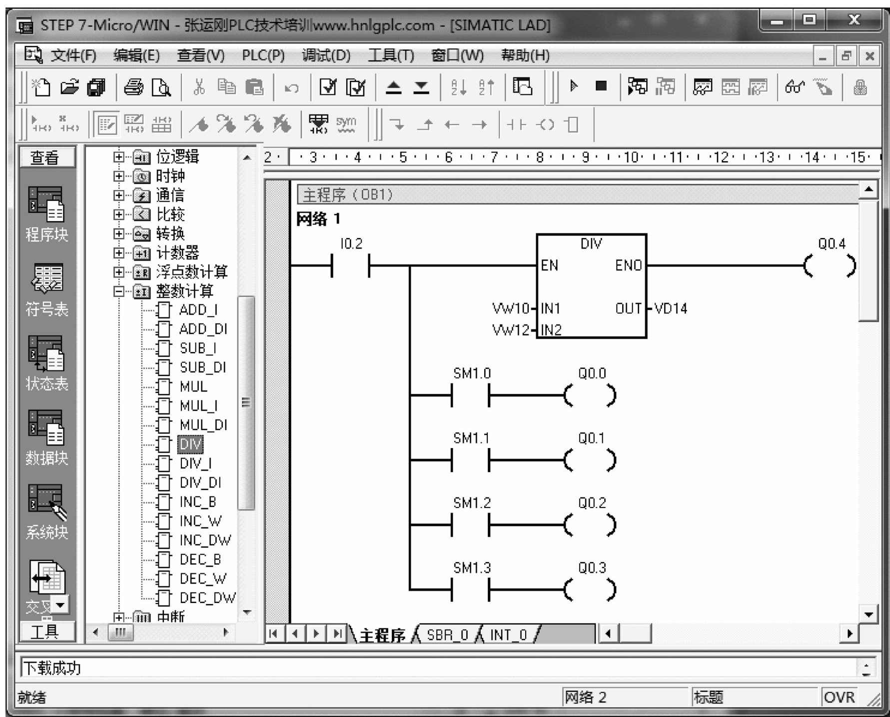


图 8-24　DIV 指令应用

问 4：在应用整数运算指令时需要注意些什么？

答：当在使用“＋”、“－”、“×”、“÷”的 16 位整数或 32 位整数四则运算指令时，需要注意如下 3 种情况：

一是间接寻址时注意不要超出寻址范围。

二是脉冲执行与连续执行是有区别的。

三是注意结果超出操作数范围；除法时，注意除数为“0”的情况。

8.2　小数运算

问：小数的特征是什么？
小数运算指令样式是怎样的？
小数运算指令基本运算规律怎样应用？
在应用小数运算指令时需要注意些什么？

问 1：小数的特征是什么？

答：在 S7-200 CPU 中，小数按照 ANSI/IEEE 754—1985 标准单精度格式表示。单精度小数占用 32 位，最高位 Bit31 是符号位，其他位是数值位。数值位分尾数段和指数段数，Bit0 ~ Bit22 共 23 位是尾数段数，Bit23 ~ Bit30 共 8 位是指数段数。小数不管是正数还是负数都是原码存放，小数见表 8-6 所示。

表 8-6　单精度小数格式

符号位	数值位											
	尾数段						指数段					
B31	B30	B29	B28	…	B24	B23	B22	B21	B20	…	B1	B0
0：正；1：负	E7	E6	E5		E1	E0	A22	A21	A20		A1	A0

问 2：小数运算指令样式是怎样的?

答：小数四则运算指令包括“＋”、“－”、“×”、“÷”这些指令，同时还包括三角函数、开方根、对数和指数指令。这些指令样式分别如图 8-25～图 8-34 所示。

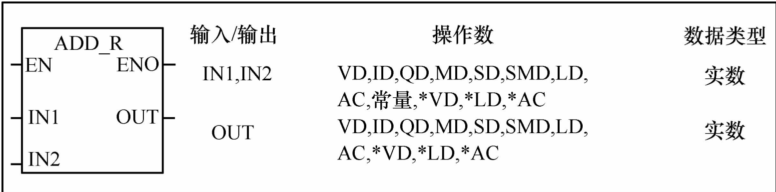


图 8-25　小数加法



图 8-26　小数减法

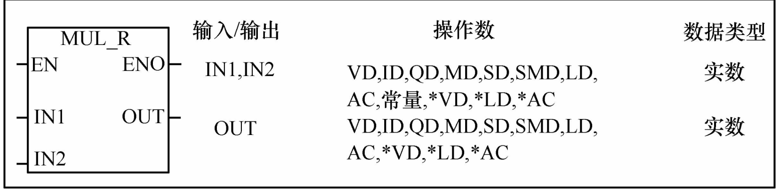


图 8-27　小数乘法

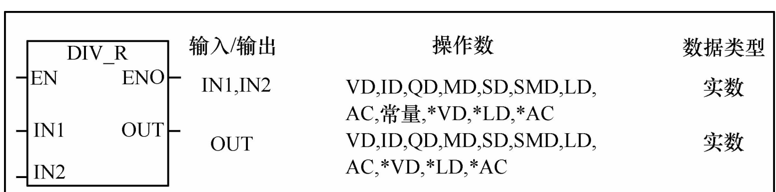


图 8-28　小数除法

SQRT	输入/输出	操作数	数据类型
EN ENO	IN	VD,ID,QD,MD,SD,SMD,LD,AC,常量,*VD,*LD,*AC	实数
IN OUT	OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	实数

图 8-29　开方根

SIN	输入/输出	操作数	数据类型
EN ENO	IN	VD,ID,QD,MD,SD,SMD,LD,AC,常量,*VD,*LD,*AC	实数
IN OUT	OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	实数

图 8-30　正弦

COS	输入/输出	操作数	数据类型
EN ENO	IN	VD,ID,QD,MD,SD,SMD,LD,AC,常量,*VD,*LD,*AC	实数
IN OUT	OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	实数

图 8-31　余弦

TAN	输入/输出	操作数	数据类型
EN ENO	IN	VD,ID,QD,MD,SD,SMD,LD,AC,常量,*VD,*LD,*AC	实数
IN OUT	OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	实数

图 8-32　正切

LN	输入/输出	操作数	数据类型
EN ENO	IN	VD,ID,QD,MD,SD,SMD,LD,AC,常量,*VD,*LD,*AC	实数
IN OUT	OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	实数

图 8-33　自然对数

EXP	输入/输出	操作数	数据类型
EN ENO	IN	VD,ID,QD,MD,SD,SMD,LD,AC,常量,*VD,*LD,*AC	实数
IN OUT	OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	实数

图 8-34　自然指数

问 3：小数运算指令基本运算规律怎样应用？

答：小数指令的规律和应用分别描述如下：

在执行 ADD_R 小数加法指令时，IN1 + IN2 = OUT，同时特殊继电器标志位会动作。

在图 8-35 所示的程序中，当 I0.2 接通，IN1 源 1 VD10 的 32 位小数加上 IN2 源 2 VD14 的 32 位小数把结果传送到 OUT 目标 VD18 32 位小数（即是 VD10 + VD14 = VD18），各个数值都是有符号数，二进制中最高位是符号位，“0”表示正数，“1”表示负数。

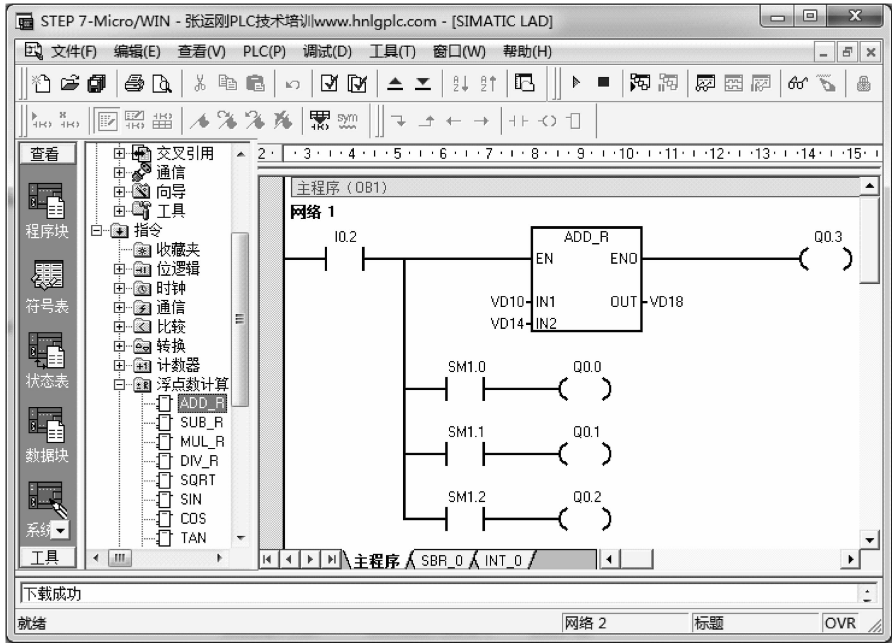


图 8-35　小数加法指令应用

当加的结果为 0，零标志位 SM1.0 会变为“1”。

当加的结果超出正数 $+1.175495\times10^{-38}$ ~ $+3.402823\times10^{38}$（$+1.175495E-38$ ~ $+3.402823E+38$）或负数 -1.175495×10^{-38} ~ $+3.402823\times10^{38}$（$-1.175495E-38$ ~ $-3.402823E+38$）的范围而非零，溢出标志位 SM1.1 会变为“1”。

当加的结果在 $-1.175495E-38$ ~ $-3.402823E+38$，负结果标志位 SM1.2 会变为“1”。

图 8-35 所示的程序调试结果见表 8-7。

表 8-7　ADD_R 调试结果

被加数 VD10	加数 VD14	和数 VD18	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	ENO Q0.3
0.0	0.0	0.0	1	0	0	1
2000.0	30201.0	32201.0	0	0	0	1
−330.0	−542.0	−872.0	0	0	1	1
$+3.402823\times10^{-38}$ ($+3.402823E+38$)	$+3.402823\times10^{38}$ ($+3.402823E+38$)	不可信	0	1	0	0
-3.402823×10^{38} ($-3.402823E+38$)	$+3.402823\times10^{38}$ ($-3.402823E+38$)	不可信	0	1	0	0

在执行 SUB_R 小数减法指令时，IN1 - IN2 = OUT，同时特殊继电器标志位会动作。

在图 8-36 所示的程序中，当 I0.2 接通，IN1 源 1 VD10 的 32 位小数减去 IN2 源 2 VD14 的 32 位小数把结果传送到 OUT 目标 VD18 32 位小数（VD10 - VD14 = VD18），各个数值都是有符号数，二进制中最高位是符号位，“0”表示正数，“1”表示负数。

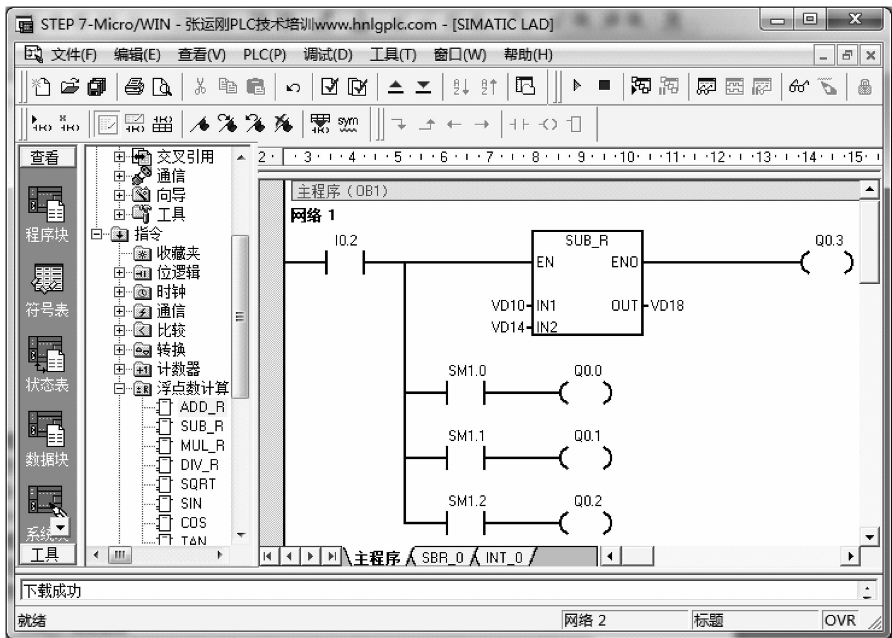


图 8-36　小数减法指令应用

当减的结果为 0，零标志位 SM1.0 会变为“1”。

当减的结果超出正数 $+1.175495\times10^{-38}$ ~ $+3.402823\times10^{38}$（+1.175495E-38 ~ +3.402823E+38）或负数 -1.175495×10^{-38} ~ $+3.402823\times10^{38}$（-1.175495E-38 ~ -3.402823E+38）的范围而非零，溢出标志位 SM1.1 会变为“1”。

当减的结果在 -1.175495E-38 ~ -3.402823E+38（-1.175495E-38 ~ -3.402823E+38），负结果标志位 SM1.2 会变为“1”。

图 8-36 所示的程序调试结果见表 8-8。

表 8-8　SUB_R 调试结果

被减数 VD10	减数 VD14	差数 VD18	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	ENO Q0.3
10.0	10.0	0.0	1	0	0	1
2000.0	-30201.0	32201.0	0	0	0	1
-330.0	542.0	-872.0	0	0	1	1
$+3.402823\times10^{38}$ (+3.402823E+38)	-3.402823×10^{-38} (-3.402823E+38)	不可信	0	1	0	0
-3.402823E+38	+3.402823E+38	不可信	0	1	0	0

在执行 MUL_R 小数乘法指令时，IN1 × IN2 = OUT （32 位小数 × 32 位小数 = 32 位小数），同时特殊继电器标志位会动作。

在图 8-37 所示的程序中，当 I0.2 接通，IN1 源 1 VD10 的 32 位小数乘以 IN2 源 2 VD14 的 32 位小数把结果传送到 OUT 目标 VD18 32 位小数（VD10 × VD14 = VD18），各个数据都是有符号数，二进制中最高位是符号位，“0”表示正数，“1”表示负数。

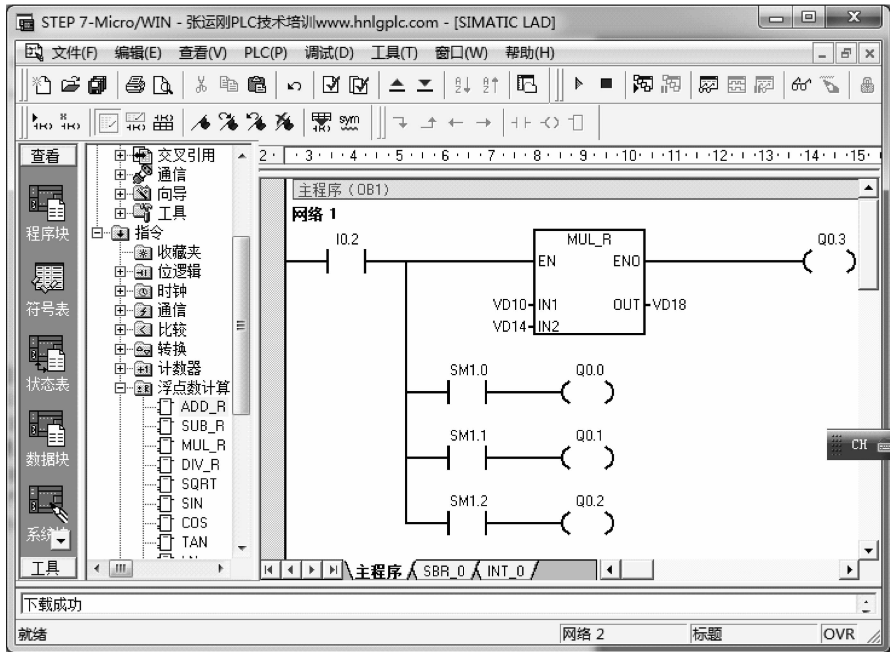


图 8-37　小数乘法指令应用

当乘的结果为 0，零标志位 SM1.0 会变为“1”。

当乘的结果超出正数 $+1.175495 \times 10^{-38} \sim +3.402823 \times 10^{-38}$（+1.175495E−38 ~ +3.402823E+38）或负数 $-1.175495 \times 10^{-38} \sim -3.402823 \times 10^{-38}$（−1.175495E−38 ~ −3.402823E+38）的范围而非零，溢出标志位 SM1.1 会变为“1”。

当乘的结果在 $-1.175495 \times 10^{-38} \sim -3.402823 \times 10^{-38}$（−1.175495E−38 ~ −3.402823E+38），负结果标志位 SM1.2 会变为“1”。

图 8-37 所示的程序调试结果见表 8-9。

表 8-9　MUL_ R 调试结果

被乘数 VD10	乘数 VD14	积数 VD18	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	ENO Q0.3
10.0	0.0	0.0	1	0	0	1
2000.0	15.0	30000.0	0	0	0	1
−330.0	2.0	−660.0	0	0	1	1
$+3.402823 \times 10^{38}$ (+3.402823E+38)	$-3.402823 \times 10^{-38}$ (−3.402823E+38)	不可信	0	1	0	0
$-3.402823 \times 10^{-38}$ (−3.402823E+38)	$+3.402823 \times 10^{38}$ (+3.402823E+38)	不可信	0	1	0	0

在执行 DIV_R 小数除法指令时，IN1 ÷ IN2 = OUT（32 位小数 ÷ 32 位小数 = 32 位小数），同时特殊继电器标志位会动作。

在图 8-38 所示的程序中，当 I0.2 接通，IN1 源 1 VD10 的 32 位小数除以 IN2 源 2 VD14 的 32 位小数把结果传送到 OUT 目标 VD18 32 位小数（VD10 ÷ VD14 = VD18），各个数据都是有符号数，二进制中最高位是符号位，“0”表示正数，“1”表示负数。

当除的结果为 0，零标志位 SM1.0 会变为“1”。

当除的结果超出正数 $+1.175495 \times 10^{-38} \sim +3.402823 \times 10^{38}$（+1.175495E－38 ~ +3.402823E＋38）或负数 $-1.175495 \times 10^{-38} \sim -3.402823 \times 10^{38}$（－1.175495E－38 ~ －3.402823E＋38）的范围而非零，溢出标志位 SM1.1 会变为“1”。

当除的结果在 $-1.175495 \times 10^{-38} \sim 3.402823 \times 10^{-38}$（－1.175495E－38 ~ －3.402823E＋38），负结果标志位 SM1.2 会变为“1”。

当除数为零，除数为零标志位 SM1.3 会变为“1”。

图 8-38 所示的程序调试结果见表 8-10。

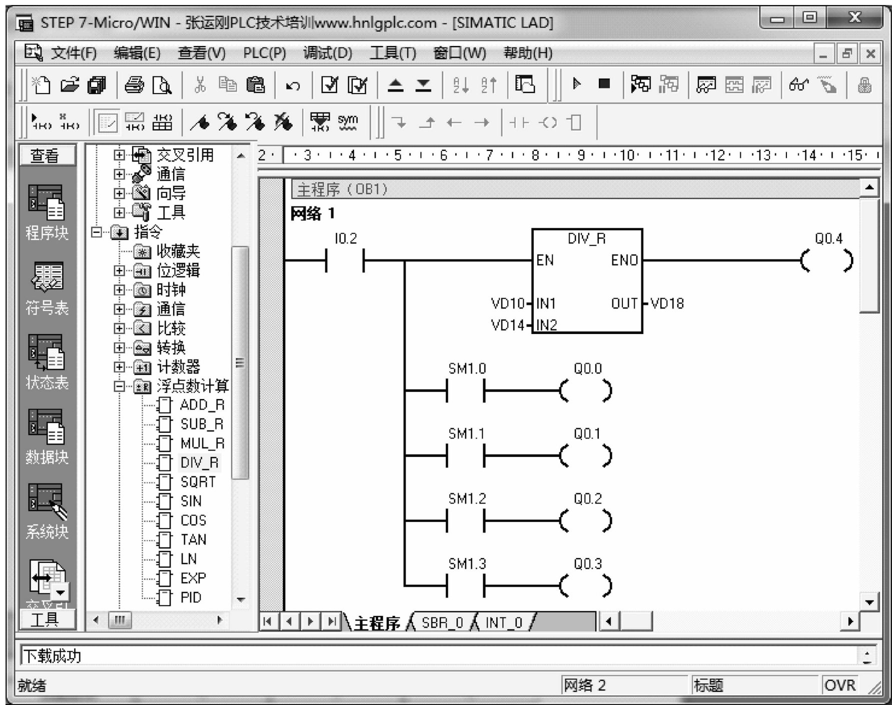


图 8-38　小数除法指令应用

表 8-10　DIV_R 调试结果

被除数 VD10	除数 VD14	商数 VD18	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	除数为零标志位 SM1.3(Q0.3)	ENO Q0.4
0.0	100.0	0.0	1	0	0	0	1
－300.0	2.0	－150.0	0	0	1	0	1
4200.0	2.0	2100.0	0	0	0	0	1
4200.0	0.0	不可信	0	0	0	1	0

SQRT 求开方根指令在输入字（IN）上求开方根，并将结果置入 OUT 指定的变量中。

当执行 SQRT 求开方根指令结果为 0 时，零标志位 SM1.0 会变为“1”。

当执行 SQRT 求开方根指令结果不在正数 +1.175495E-38 ~ +3.402823E+38 的范围或非零，溢出标志位 SM1.1 会变为“1”。

如图 8-39 所示程序，当 I0.2 接通时，输入与输出、特殊继电器标志位及 ENO 的状态见表 8-11。

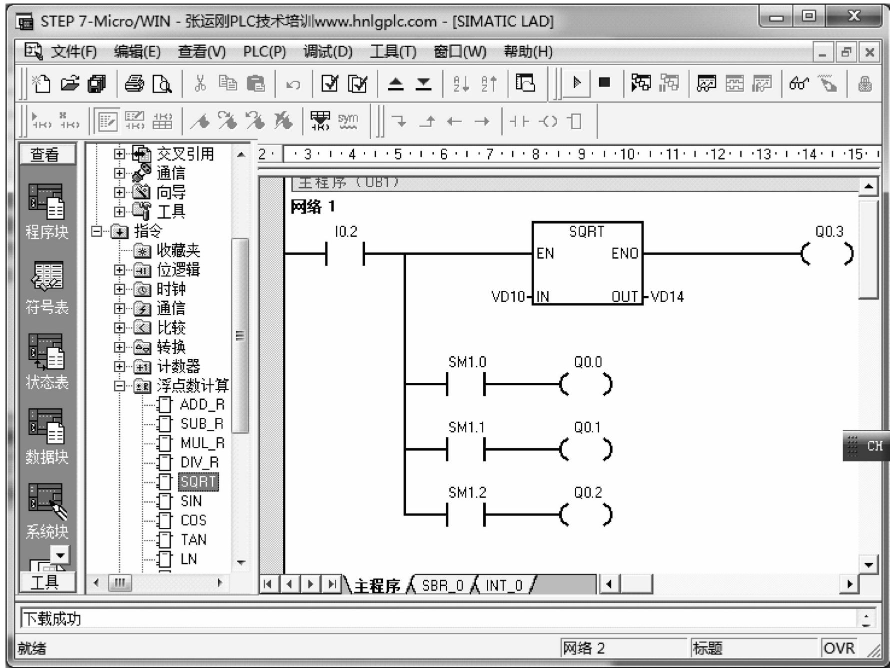


图 8-39　SQRT 开方根指令应用

表 8-11　SQR 调试结果

VD10 数值	VD14 数值	零标志位 SM1.0（Q0.0）	溢出标志位 SM1.1（Q0.1）	ENO（Q0.3）
0.0	0.0	1	0	1
负数	不可信	0	1	0
900.0	30.0	0	0	1
1.175495E-38	1.084202E-19	0	0	1
3.402823E+38	1.844674E+19	0	0	1

三角函数 IN 源需要输入的小数是弧度（以下称为 RAD），RAD = 角度数 ×π/180 或经验公式 RAD = 角度数 ×1.745329E-2。圆一周的角度数是 360.0°。

正弦（SIN）指令对 IN 源角度值进行三角正弦运算，并将结果放置在 OUT 中。

余弦（COS）指令对 IN 源角度值进行三角余弦运算，并将结果放置在 OUT 中。

正切（TAN）指令对 IN 源角度值进行三角正切运算，并将结果放置在 OUT 中。

当运算的结果为 0，零标志位 SM1.0 会变为“1”。

当运算的结果不在正数 $+1.175495\times10^{-38}$ ~ $+3.402823\times10^{38}$（+1.175495E−38 ~ +3.402823E+38）或负数 -1.175495×10^{-38} ~ -3.402823×10^{38}（−1.175495E−38 ~ −3.402823E+38）的范围，或者不为零，溢出标志位 SM1.1 会变为“1”。

当运算的结果在 -1.175495×10^{-38} ~ -3.402823×10^{38}（−1.175495E−38 ~ −3.402823E+38），负结果标志位 SM1.2 会变为“1”。

在图 8-40 所示的程序中，当 I0.2 接通，VD100 的度数变换成角度传送到 VD108，然后求 VD108 正弦值，传送到 VD112 中。

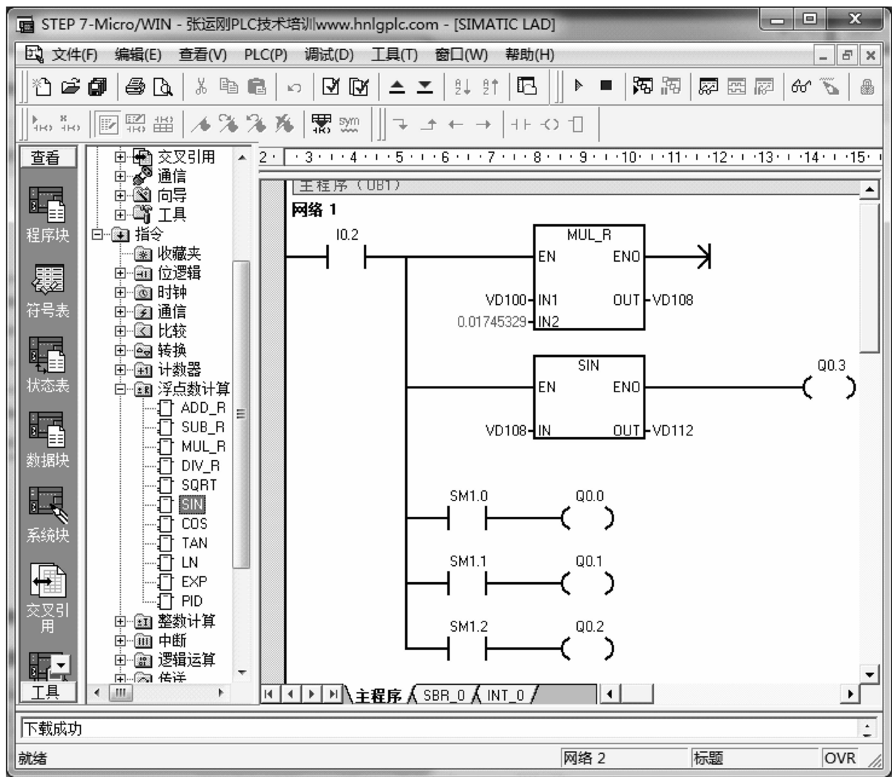


图 8-40　SIN 三角正弦指令例

在图 8-41 所示的程序中，当 I0.2 接通，VD100 的度数变换成角度传送到 VD108，然后求 VD108 余弦值结果传送到 VD112 中。

在图 8-42 所示的程序中，当 I0.2 接通，VD100 的度数变换成角度传送到 VD108，然后求 VD108 正切值，传送到 VD112 中。

LN 自然对数指令对 IN 中的数值进行自然对数计算，并将结果置于 OUT 中。

当运算的结果为 0，零标志位 SM1.0 会变为“1”。

自然对数指令如图 8-43 所示。

当运算的结果不在正数 $+1.175495\times10^{-38}$ ~ $+3.402823\times10^{38}$（+1.175495E−38 ~ +3.402823E+38）或负数 -1.175495×10^{-38} ~ -3.402823×10^{38}（−1.175495E−38 ~ −3.402823E+38）或零的范围，溢出标志位 SM1.1 会变为“1”。

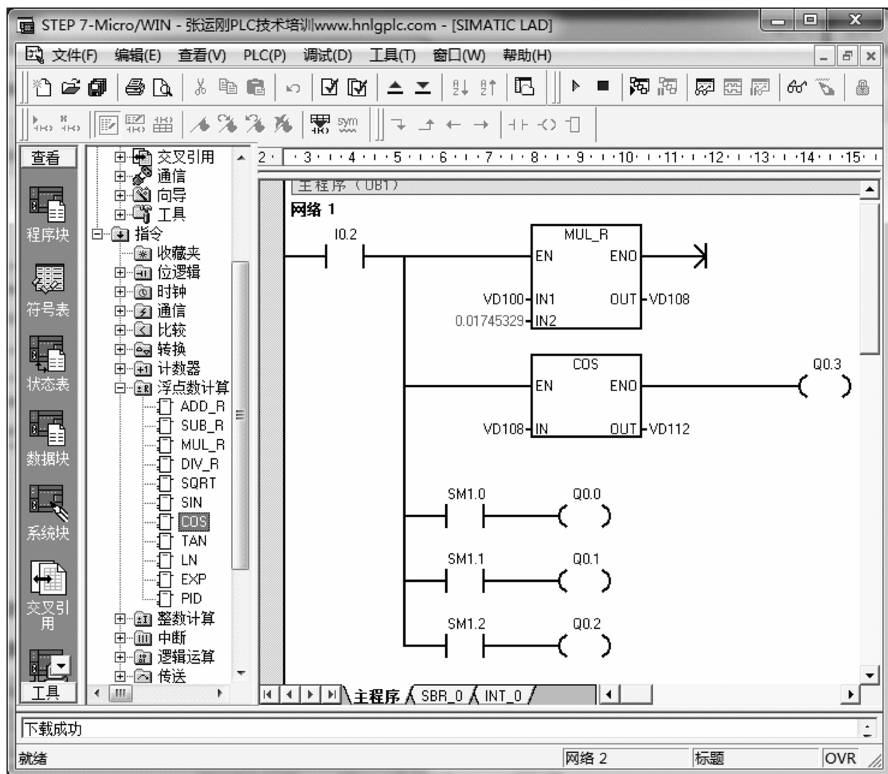


图 8-41　COS 三角余弦指令应用

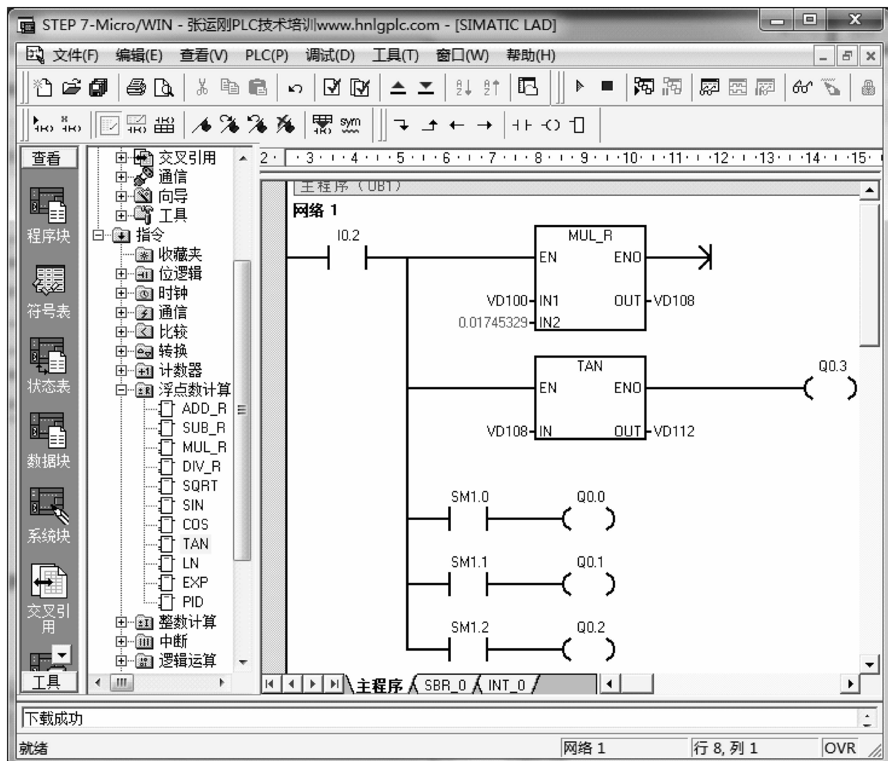


图 8-42　TAN 三角正切指令应用

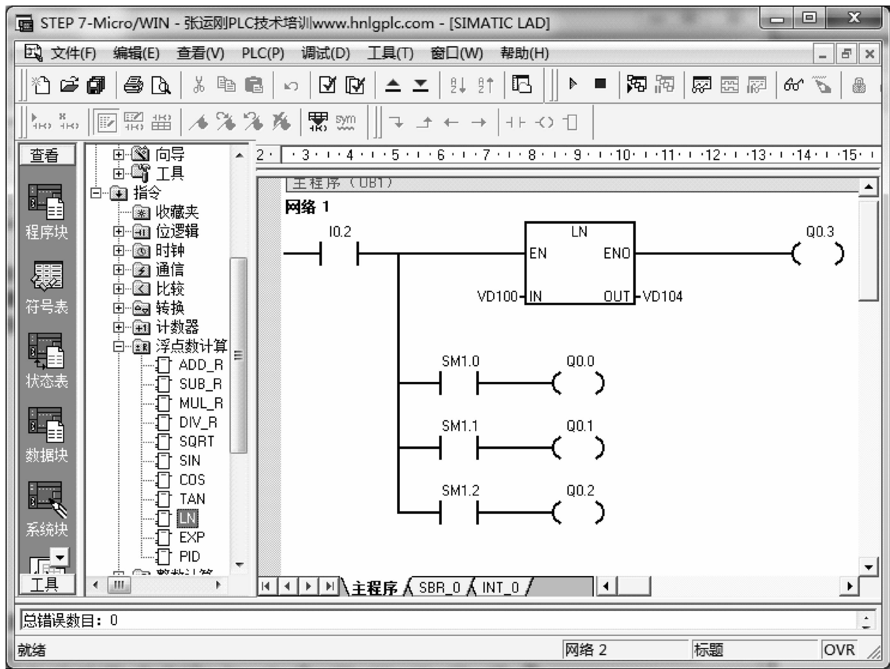


图 8-43　LN 自然对数指令应用

当运算的结果在 -1.17549510^{38} ~ -3.40282310^{38}（$-1.175495E-38$ ~ $-3.402823E+38$），负结果标志位 SM1.2 会变为“1”。

要求计算数学公式“$\lg X=Y$”，即是求以 10 为底 X 的对数等于 Y，使用 PLC 编程实现如图 8-44 所示。其中 VD100 代表变量 X，VD108 代表变量 Y。

EXP 自然指数指令对 IN 中的数值进行自然指数计算，并将结果置于 OUT 中。

当运算的结果为 0，零标志位 SM1.0 会变为“1”。

当运算的结果不在正数 $+1.175495\times10^{-38}$ ~ $+3.402823\times10^{38}$（$+1.175495E-38$ ~ $+3.402823E+38$）或负数 -1.175495×10^{-38} ~ -3.402823×10^{38}（$-1.175495E-38$ ~ $-3.402823E+38$）或零的范围，溢出标志位 SM1.1 会变为“1”。

当运算的结果在 -1.175495×10^{-38} ~ 3.402823×10^{38}（$-1.175495E-38$ ~ $-3.402823E+38$），负结果标志位 SM1.2 会变为“1”。

EXP 自然指数指令应用如图 8-45 所示。要求计算数学公式“$e^X=Y$”，即是求以 e 为底 X 次方等于 Y，使用 PLC 编程实现如图 8-46 所示。其中 VD100 代表变量 X，VD104 代表变量 Y。

要求计算数学公式“$X1^{X2}=Y$”，即是求 X1 的 X2 次方等于 Y，使用 PLC 编程实现如图 8-46 所示。其中 VD100 代表变量 X1，VD104 代表变量 X2，VD108 代表变量 Y。

要求计算数学公式“$\sqrt[X3]{X1^{X2}}$”，使用 PLC 编程实现如图 8-47 ~ 图 8-49 所示。其中 VD100 代表变量 X1，VD104 代表变量 X2，VD108 代表变量 X3，VD112 代表结果。

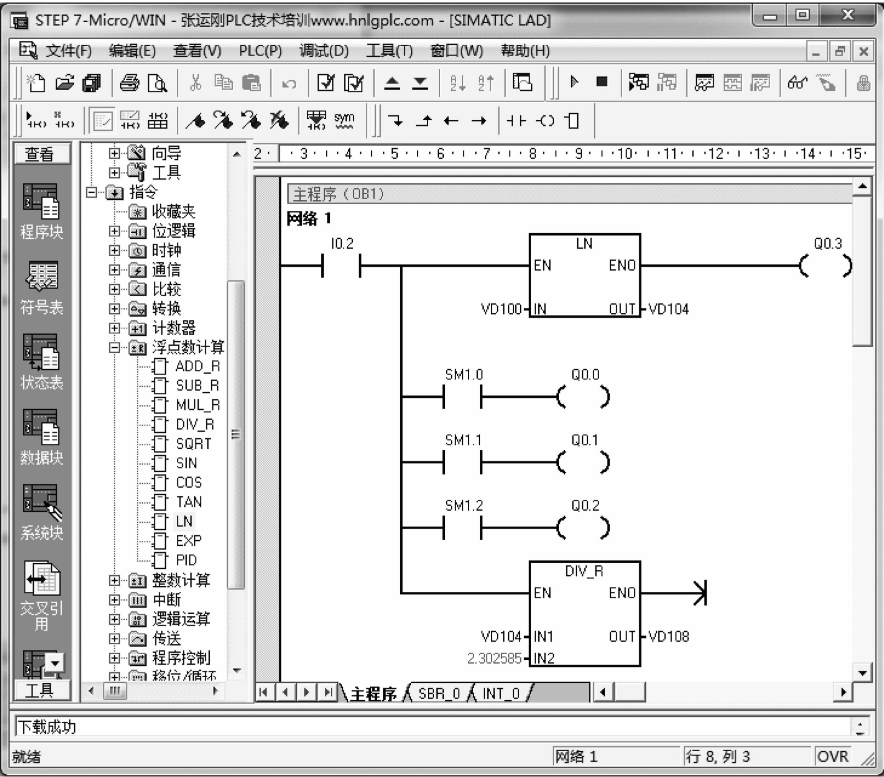


图 8-44　Y = lgX 的计算方法

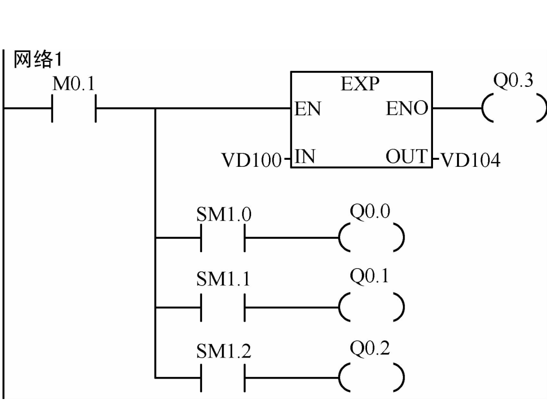


图 8-45　EXP 自然指数指令应用

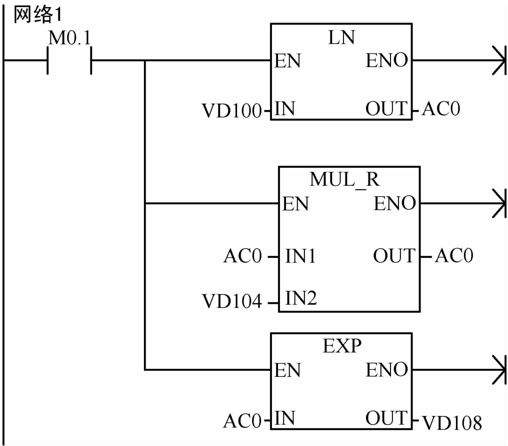


图 8-46　$X1^{X2}$ = Y 的计算方法

特例 1：5 的立方 = 5^3 = EXP[3 * LN(5)] = 125，如图 8-47 所示。

特例 2：125 的立方根 = 125^(1/3) = EXP[1/3 * LN(125)] = 5，如图 8-49 所示。

特例 3：5 的立方的平方根 = 5^(3/2) = EXP[3/2 * LN(5)] = 11.18034，如图 8-49 所示。

问 4：在应用小数运算指令时需要注意些什么？

答：当在使用小数数运算指令时，需要注意如下几种情况：

一是间接寻址时注意不要超出寻址范围。

二是脉冲执行与连续执行是有区别的。

三是注意结果超出操作数范围；除法时，注意除数为“0”的情况。

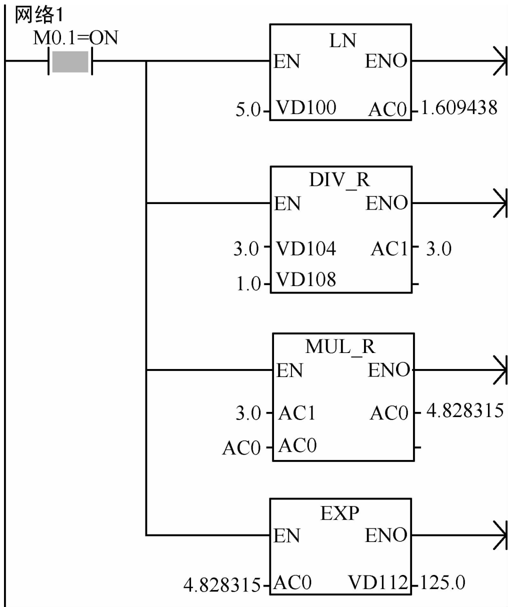


图 8-47　求 5 的立方程序

四是三角函数的指令数值单位为弧度，而不是一般人平时习惯所有的角度数。

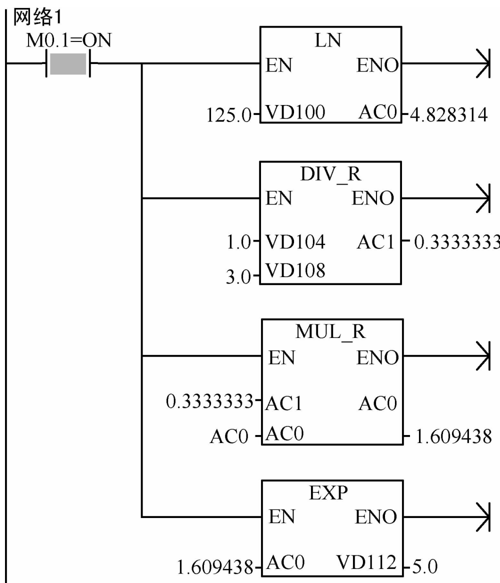


图 8-48　求 125 的立方根程序

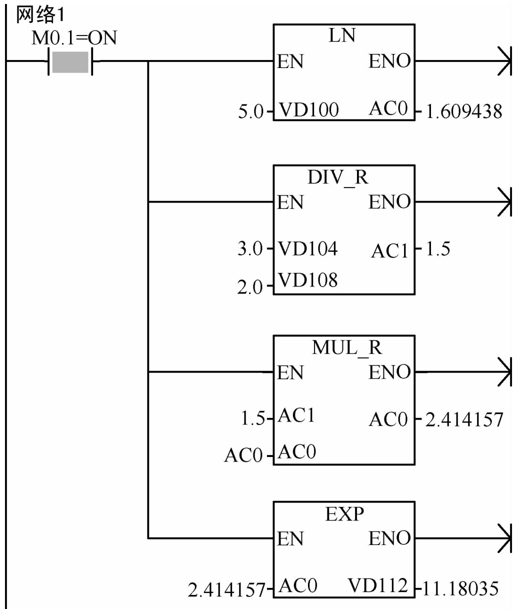


图 8-49　求 5 的立方的平方根程序

8.3　数值类型转换 B←→I←→DI←→R

问：能否计算有 8 位字节、16 位整数、32 位整数和小数的混合运算？ 数值转换指令样式是怎样的？ 这些转换指令基本规律怎样？ 在应用数值转换指令时需要注意些什么？

问 1：能否计算有 8 位字节、16 位整数、32 位整数和小数的混合运算？

答：S7-200PLC 是可以计算有 8 位字节、16 位整数、32 位整数和小数的混合运算，在进行运算之前需要使用数值转换指令，转换成相同的数值类型即可进行计算。

问 2：数值转换指令样式是怎样的？

答：这些转换指令包括 BTI 字节转换整数、ITD 整数转换双整数、DTR 双整数转换小数、ROUND 四舍五入取双整数、TRUNC 舍小数取双整数、DTI 双整数转换整数、ITB 整数转换字节，这些转换指令样式如图 8-50 ~ 图 8-56 所示。

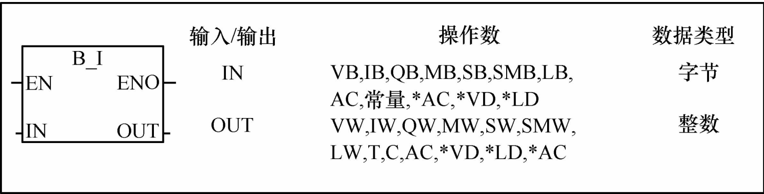


图 8-50　字节转换成字指令样式



图 8-51　字转换成双字指令样式



使ENO=0的错误条件:
0006间接地址

图 8-52　双字转换成小数指令样式



图 8-53　小数转换成双字指令样式 1



使ENO=0的错误条件:
0006间接地址
SM1.1溢出或非法数值

影响特殊继电器位:
SM1.1(溢出)

图 8-54　小数转换成双字指令样式 2



图 8-55　双字转换成字指令样式



图 8-56　字转换成字节指令样式

问 3：这些转换指令基本规律怎样？

答：数值转换指令规律分别描述如下：

BTI 字节转换整数指令将字节（IN）数值转换成整数值，并将结果传送到 OUT 指定的变量中。因为字节不带符号，所以无符号扩展。

ITD 整数转换双整数指令将整数值（IN）转换成双整数值，并将结果传送到 OUT 指定的变量中，符号也被扩展。

DTR 双整数转换小数指令将 32 位带符号整数（IN）转换成 32 位小数，并将结果传送到 OUT 指定的变量中。

ROUND 四舍五入取双整指令将小数值（IN）转换成双整数值，并将结果传送到 OUT 指定的变量中。如果小数部分等于或大于 0.5，则进位为整数；如果小数部分小于 0.5，则舍去小数部分取整数。

TRUNC 舍去小数部分取整指令将 32 位小数（IN）转换成 32 位双整数，并将结果的整数部分传送到 OUT 指定的变量中。只有小数的整数部分被转换，小数部分被舍弃。如果要转换的数值为无效小数或数值过大，无法在输出中表示，则溢出特殊继电器标志位会为“1”，输出保留原来不变。

DTI 双整数转换整数指令将双整数值（IN）转换成整数值，并将结果传送到 OUT 指定的变量中。IN 数值在 -32768 ~ 32767 范围被正确转换；如果 IN 数值小于 -32768 或大于 32767 的数值不能转换，输出保留原来不变，溢出特殊继电器标志位会为“1”。

ITB 整数转换字节指令将整数值（IN）转换成字节数值，并将结果传送到 OUT 指定的变量中。IN 数值在 0 ~ 255 范围被正确转换；IN 数值如果大于 255 或为负数不能转换，结果保留原来不变，溢出特殊继电器标志位会为“1”。

BTI、ITD、DTR 指令及调试状态如图 8-57 所示。

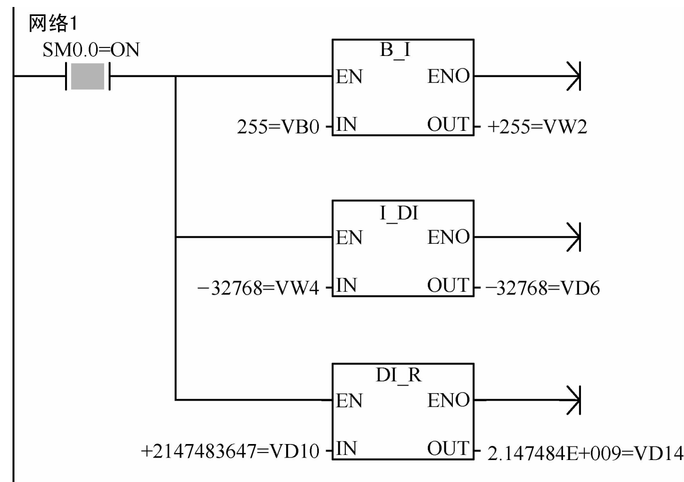


图 8-57　BTI、ITD、DTR 指令

图 8-58 所示为 ITB 指令应用例，调试结果见表 8-12。

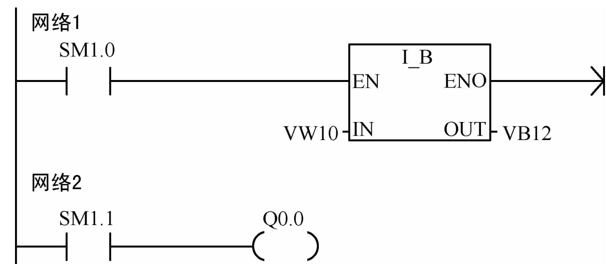


图 8-58　ITB 指令

表 8-12　ITB 指令调试结果

IN 中数值范围	VW10	VB12	SM1. 1	ENO
0 ~ 255	140	140	0	1
256 ~ 32 767	300	保留原来不变	1	0
-1 ~ -32 768	-300	保留原来不变	1	0

图 8-59 所示为 DTI 指令应用例，调试结果见表 8-13。

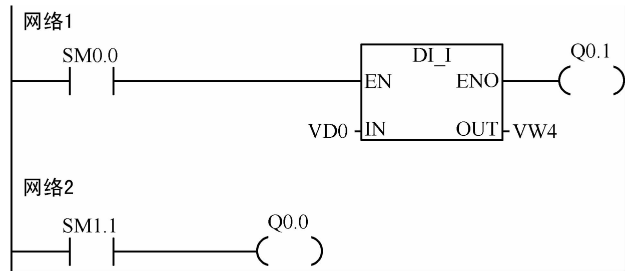


图 8-59　DTI 指令

表 8-13　DTI 指令调试结果

IN 中数值范围	VD0	VW4	SM1. 1	ENO
-32 768 ~ 32 767	140	140	0	1
小于 -32 768	-32 769	保留原来不变	1	0
大于 32 767	32 768	保留原来不变	1	0

ROUND 四舍五入取整指令如图 8-60 所示。

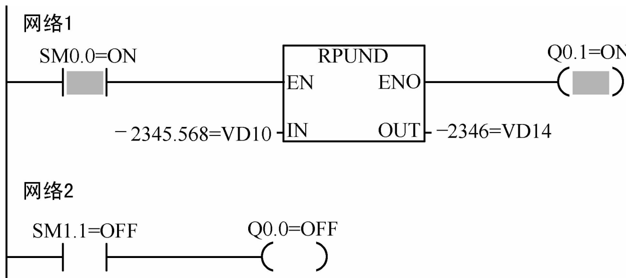


图 8-60　ROUND 指令

TRUNC 舍去小数取整指令如图 8-61 所示。

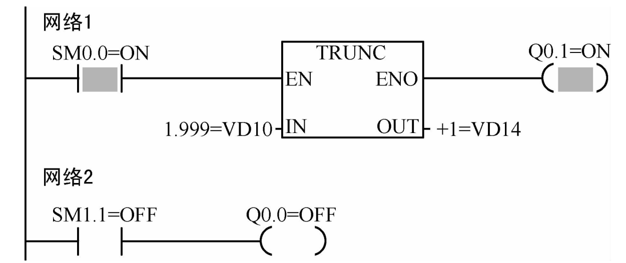


图 8-61　TRUNC 指令

问 4：在应用数值转换指令时需要注意些什么？

答：当在使用数值转换指令时，需要注意如下几种情况：

一是间接寻址时注意不要超出寻址范围。

二是注意结果超出操作数范围，特别是使用 DTI 和 ITB 指令需要特别注意。

三是在混合运算过程中，注意数值的一致性和结果超出范围。

8.4　BCD 码和七段码转换

> 问：以前没有触摸屏人机界面时能否实现人机界面功能？
> 　　人机界面数据转换指令样式是怎样的？
> 　　人机界面数据转换指令基本规律怎样？
> 　　在应用人机界面数据转换指令时需要注意些什么？

问 1：以前没有触摸屏人机界面时能否实现人机界面功能？

答：以前没有触摸屏人机界面时也能实现人机界面功能，输入数值使用 BCD _ I 指令把二进制编码的十进制数转换成整数，显示数值用 I _ BCD 指令把整数转换成二进制编码的十进制数，有些场合也会使用 SEG 指令把数值现转成十六进制数。

问 2：人机界面数据转换指令样式是怎样的？

答：BCD _ I、I _ BCD 和 SEG 指令样式如图 8-62 ~ 图 8-64 所示。



图 8-62　BCD _ I 指令样式



使ENO=0的错误条件
0006间接地址
SM1.6无效BCD数值

特殊继电器标志位
SM1.6(无效BCD)

图 8-63　I _ BCD 指令样式



使ENO=0的错误条件
0006间接地址

图 8-64　SEG 指令样式

问 3：人机界面数据转换指令基本规律怎样？

答：人机界面数据转换指令基本规律分别描述如下：

BCD _ I 指令，BCD 码转换整数指令将二进制编码的十进制（IN）数值转换成整数，并将结果载入 OUT 指定的变量中。IN 的有效范围是 0 ~ 9999 的 BCD 码。如果 IN 的有效范围超出 0 ~ 9999 的 BCD 码，执行 BCD _ I 指令时 SM1.6 会为“1”。

I _ BCD 指令，整数转换 BCD 码指令将输入整数值（IN）转换成二进制编码的十进制数，并将结果传送到 OUT 指定的变量中。IN 的有效范围是 0 ~ 9999 的 BCD 码。如果 IN 的有效范围超出 0 ~ 9999 的 BCD 码，执行 I _ BCD 指令 SM1.6 会为“1”。

SEG 指令，是转换成七段码指令将 IN 中二进制数的低四位译成七段码传送到目标（字节）低八位中。

BCD _ I 指令应用例，如果需要使用数字开关向 PLC 输入数值到 VW12，接线示意图如图 8-65 所示，编写 PLC 程序如图 8-66 所示。当数字开关显示“2794”时，监控 VW12 的数值也是“2794”。当出现非 BCD 码的数值时，VW12 的数值保持原来的不变，SM1.6 会为“1”。

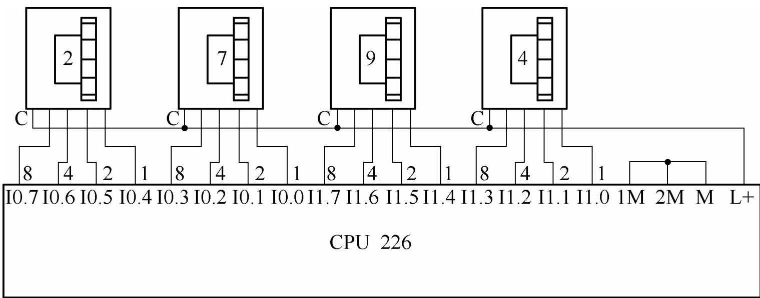


图 8-65　BCD _ I 输入接线

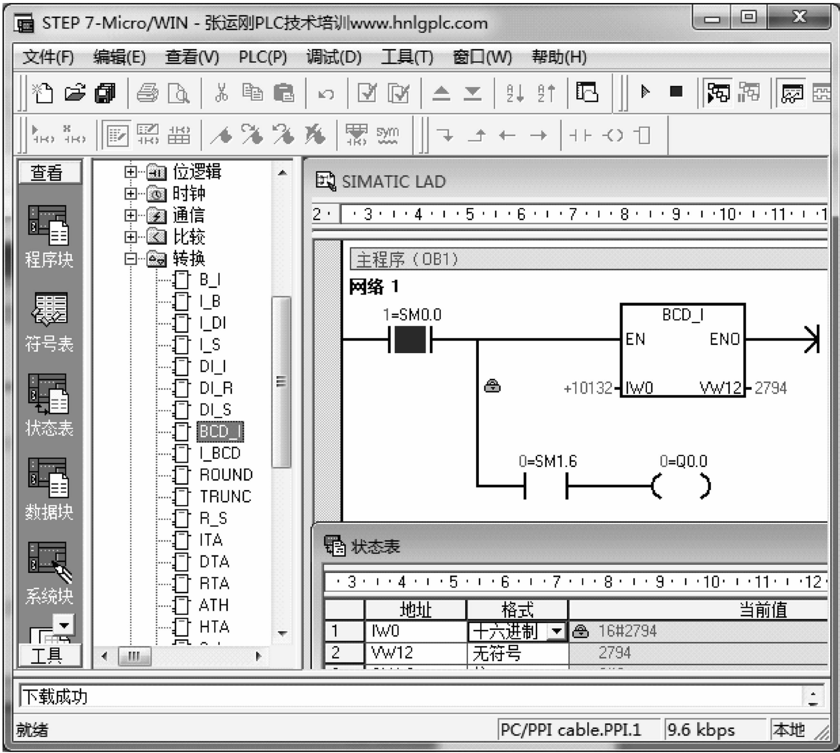


图 8-66　BCD _ I 指令应用程序

BCD 码数字开关引脚的关系见表 8-14 所示。

表 8-14　BCD 码数字开关的数值引脚关系

	数字	8	4	2	1
	0	0	0	0	0
	1	0	0	0	1
	2	0	0	1	0
	3	0	0	1	1
	4	0	1	0	0
	5	0	1	0	1
	6	0	1	1	0
	7	0	1	1	1
	8	1	0	0	0
	9	1	0	0	1
非 BCD 码数值	A	1	0	1	0
	B	1	0	1	1
	C	1	1	0	0
	D	1	1	0	1
	E	1	1	1	0
	F	1	1	1	1
说明：8、4、2、1 引脚为“1”相当于与“C”公共引脚是接通的					

BCD 码输出接线如图 8-67 所示，实现把 VW12 数值传送到 QW0 中驱动 BCD 显示管显示程序如图 8-68 所示，当 VW12 的数值为“2794”时，外面的 BCD 码显示管显示“2794”。

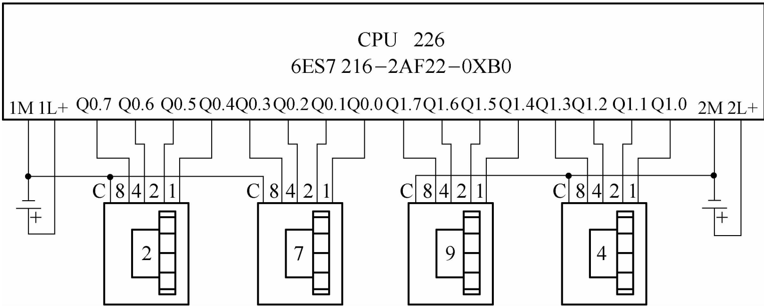


图 8-67　BCD 码输出接线

在图 8-68 中，当 VW12 的数字超出 0 ~ 9999 的 BCD 码显示范围时，执行 I _ BCD 指令，QW0 的输出保持原来不变，SM1. 6 会为“1”。

在图 8-69 所示的程序中，IN 源 VB10 中二进制数的低四位数值译成七段码传送到目标低八位中。

七段码译码规律见表 8-15。

当目标为 Q 时，驱动七段显示器的接线如图 8-70 所示。如果七段显示器是共阳极七段码显示器（如图 8-71 所示），选择 PLC 的输出为 NPN 输出；如果七段显示器是共阴极七段码显示器（如图 8-72 所示），选择 PLC 的输出为 PNP 输出。

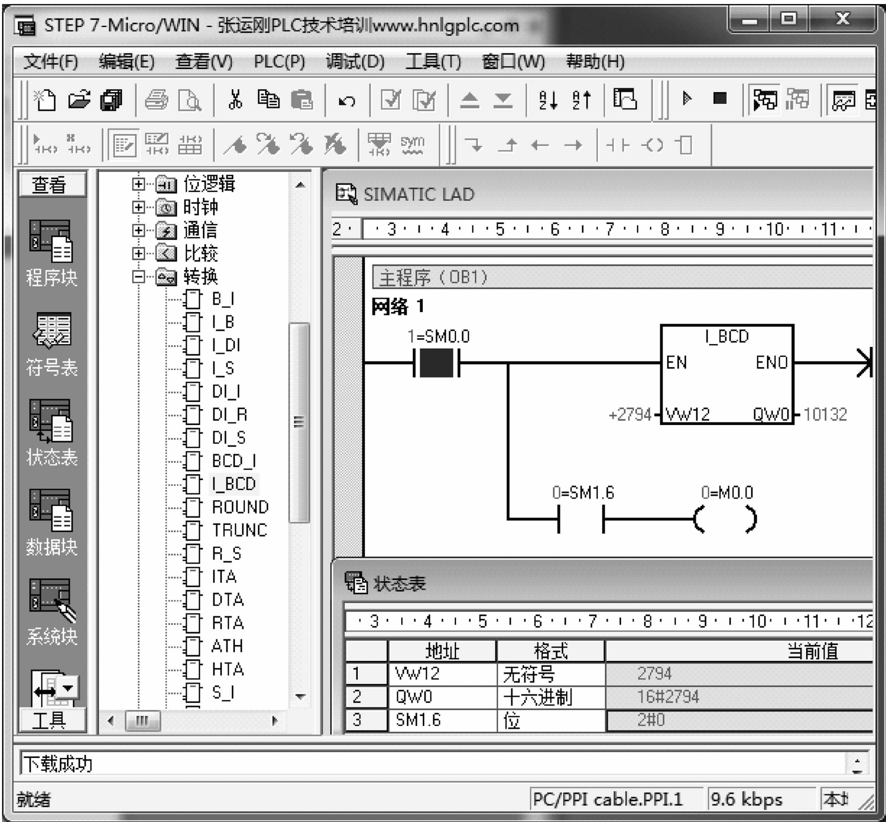


图 8-68　I _ BCD 指令 BCD 码输出程序

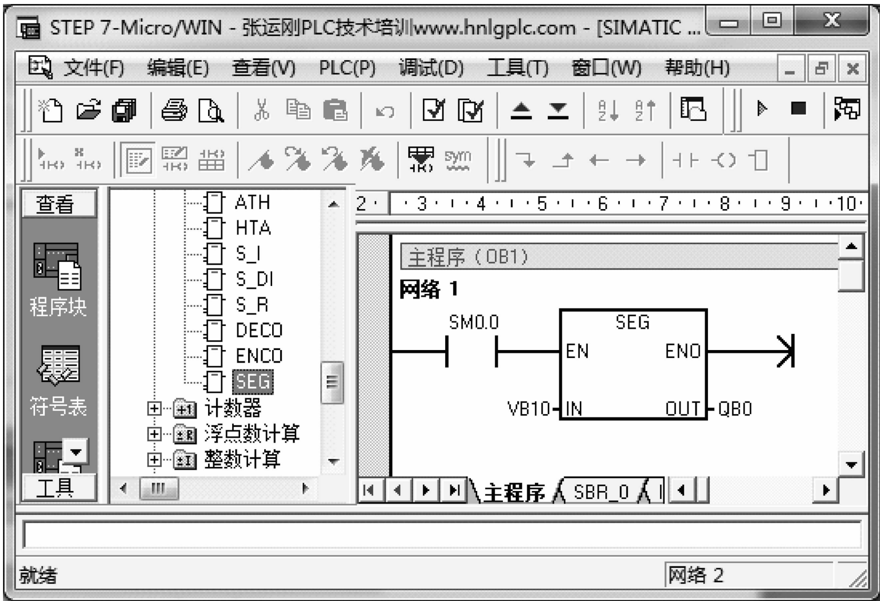


图 8-69　SEG 指令七段码输出程序

表 8-15　SEG 七段码显示译码表

IN 源			OUT 译码输出(目标)						
十进制数	十六进制数	二进制数	g	f	e	d	c	b	a
			Q0. 6	Q0. 5	Q0. 4	Q0. 3	Q0. 2	Q0. 1	Q0. 0
0	0	0000	0	1	1	1	1	1	1
1	1	0001	0	0	0	0	1	1	0
2	2	0010	1	0	1	1	0	1	1
3	3	0011	1	0	0	1	1	1	1
4	4	0100	1	1	0	0	1	1	0
5	5	0101	1	1	0	1	1	0	1
6	6	0110	1	1	1	1	1	0	1
7	7	0111	0	1	0	0	1	1	1
8	8	1000	1	1	1	1	1	1	1
IN 源			OUT 译码输出(目标)						
十进制数	十六进制数	二进制数	g	f	e	d	c	b	a
			Q0. 6	Q0. 5	Q0. 4	Q0. 3	Q0. 2	Q0. 1	Q0. 0
9	9	1001	1	1	0	1	1	1	1
10	A	1010	1	1	1	0	1	1	1
11	B	1011	1	1	1	1	1	0	0
12	C	1100	0	1	1	1	0	0	1
13	D	1101	1	0	1	1	1	1	0
14	E	1110	1	1	1	1	0	0	1
15	F	1111	1	1	1	0	0	0	1

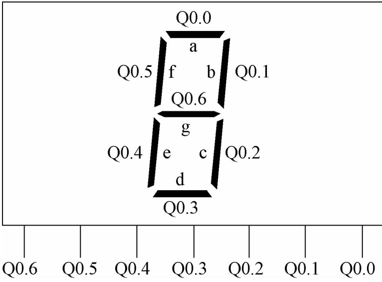


图 8-70　七段码显示器

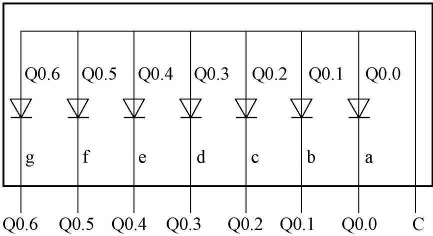


图 8-71　共阳极七段码显示器内部结构示意图

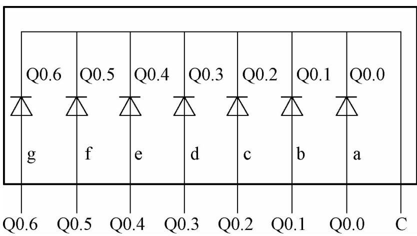


图 8-72　共阴极七段码显示器内部结构示意图

问 4：在应用人机界面数据转换指令时需要注意些什么?

答：当在使用人机界面转换指令时，需要注意如下几种情况：

一是间接寻址时注意不要超出寻址范围。

二是注意结果超出操作数范围，特别是使用 I _ BCD 和 BCD _ I 指令时需要特别注意。

8.5　字符（串）转换

> 问：S7-200 CPU 支持字符功能吗？支持字符串功能吗？支持中文字符串吗？
> 这些字符和字符串指令样式是怎样的？
> 字符和字符串指令基本规律怎样？
> 在应用字符和字符串指令时需要注意些什么？

问 1：S7-200 CPU 支持字符功能吗？支持字符串功能吗？支持中文字符串吗？

答：S7-200 CPU 支持 ASCII 字符的有效范围为 ASCII 32 ~ ASCII 255，但不包括 DEL（删除）字符、单引号字符和双引号字符。S7-200 CPU 支持字符串功能，也支持中文字符串。

问 2：这些字符和字符串指令样式是怎样的？

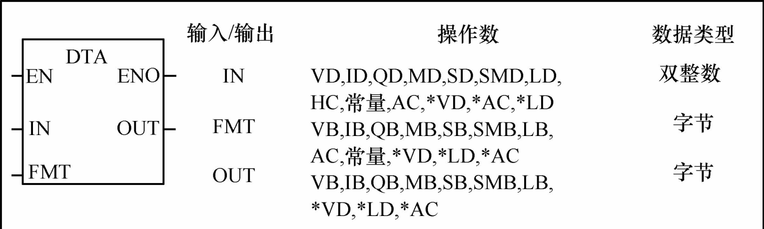
答：这些字符/字符串指令样式描述如下：

1）字符转换指令包括：ITA（16 位整数转 ASCII 字符）、DTA（32 位整数转 ASCII 字符）、RTA（小数转 ASCII 字符）、HTA（十六进制数转 ASCII 字符）和 ATH（ASCII 字符转十六进制数）指令。其指令样式分别如图 8-73 ~ 图 8-77 所示。



输入/输出	操作数	数据类型
IN	VW,IW,QW,MW,SW,SMW,LW,AIW,T,C,AC,常量,*VD,*LD,*AC	整数
FMT	VB,IB,QB,MB,SB,SMB,LB,AC,常量,*VD,*LD,*AC	字节
OUT	VB,IB,QB,MB,SB,SMB,LB,*VD,*LD,*AC	字节

图 8-73　ITA（16 位整数转 ASCII 字符）



输入/输出	操作数	数据类型
IN	VD,ID,QD,MD,SD,SMD,LD,HC,常量,AC,*VD,*AC,*LD	双整数
FMT	VB,IB,QB,MB,SB,SMB,LB,AC,常量,*VD,*LD,*AC	字节
OUT	VB,IB,QB,MB,SB,SMB,LB,*VD,*LD,*AC	字节

图 8-74　DTA（32 位整数转 ASCII 字符）



输入/输出	操作数	数据类型
IN	VD,ID,QD,MD,SD,SMD,LD,AC,常量,*VD,*LD,*AC	实数
FMT	VB,IB,QB,MB,SB,SMB,LB,AC,常量,*VD,*LD,*AC	字节
OUT	VB,IB,QB,MB,SB,SMB,LB,*VD,*LD,*AC	字节

图 8-75　RTA（小数转 ASCII 字符）



输入/输出	操作数	数据类型
IN,OUT	VB,IB,QB,MB,SB,SMB,LB, *VD,*AC,*LD	字节
LEN	VB,IB,QB,MB,SB,SMB,LB, AC,常量,*VD,*LD,*AC	字节

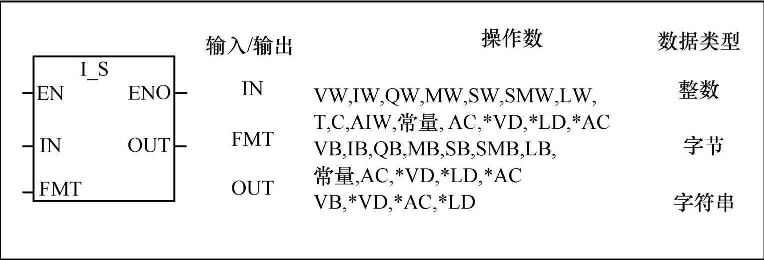
图 8-76　HTA（十六进制数转 ASCII 字符）



输入/输出	操作数	数据类型
IN,OUT	VB,IB,QB,MB,SB,SMB,LB, *VD,*AC,*LD	字节
LEN	VB,IB,QB,MB,SB,SMB,LB, AC,常量,*VD,*LD,*AC	字节

图 8-77　ATH（ASCII 字符转十六进制数）

2）字符串转换指令包括：ITS（16 位整数转 ASCII 字符串）、DTS（32 位整数转 ASCII 字符串）、RTS（小数转 ASCII 字符串）、STR（ASCII 字符串转小数）、STD（ASCII 字符串转 32 位整数）和 STI（ASCII 字符串转 16 位整数）指令。其指令样式分别如图 8-78 ~ 图 8-83 所示。



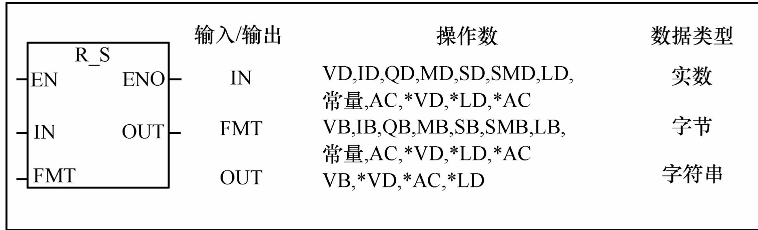
输入/输出	操作数	数据类型
IN	VW,IW,QW,MW,SW,SMW,LW, T,C,AIW,常量, AC,*VD,*LD,*AC	整数
FMT	VB,IB,QB,MB,SB,SMB,LB, 常量,AC,*VD,*LD,*AC	字节
OUT	VB,*VD,*AC,*LD	字符串

图 8-78　ITS（16 位整数转 ASCII 字符串）



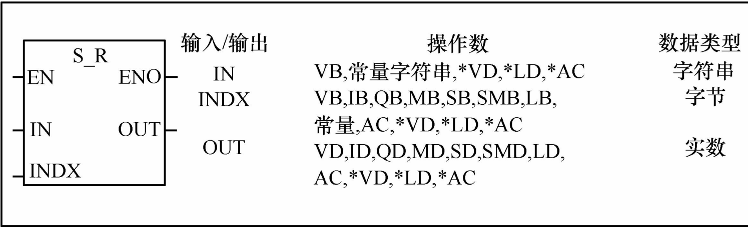
输入/输出	操作数	数据类型
IN	VD,ID,QD,MD,SD,SMD,LD,HC, 常量,AC,*VD,*AC,*LD	双整数
FMT	VB,IB,QB,MB,SB,SMB,LB, 常量,AC,*VD,*LD,*AC	字节
OUT	VB,*VD,*AC,*LD	字符串

图 8-79　DTS（32 位整数转 ASCII 字符串）



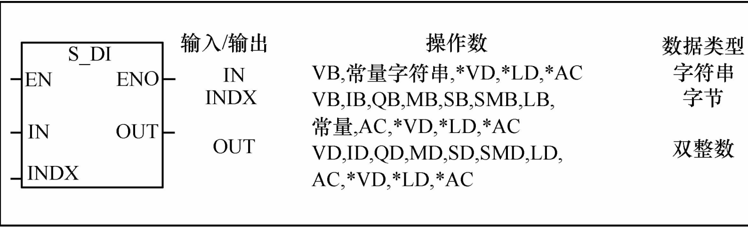
输入/输出	操作数	数据类型
IN	VD,ID,QD,MD,SD,SMD,LD, 常量,AC,*VD,*LD,*AC	实数
FMT	VB,IB,QB,MB,SB,SMB,LB, 常量,AC,*VD,*LD,*AC	字节
OUT	VB,*VD,*AC,*LD	字符串

图 8-80　RTS（小数转 ASCII 字符串）



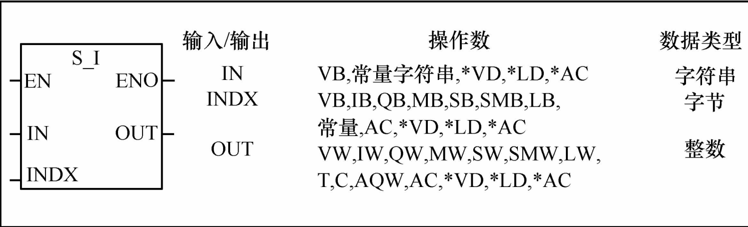
输入/输出	操作数	数据类型
IN	VB,常量字符串,*VD,*LD,*AC	字符串
INDX	VB,IB,QB,MB,SB,SMB,LB,常量,AC,*VD,*LD,*AC	字节
OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	实数

图 8-81　STR（ASCII 字符串转小数）



输入/输出	操作数	数据类型
IN	VB,常量字符串,*VD,*LD,*AC	字符串
INDX	VB,IB,QB,MB,SB,SMB,LB,常量,AC,*VD,*LD,*AC	字节
OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	双整数

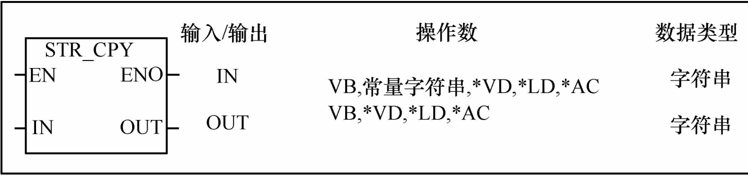
图 8-82　STD（ASCII 字符串转 32 位整数）



输入/输出	操作数	数据类型
IN	VB,常量字符串,*VD,*LD,*AC	字符串
INDX	VB,IB,QB,MB,SB,SMB,LB,常量,AC,*VD,*LD,*AC	字节
OUT	VW,IW,QW,MW,SW,SMW,LW,T,C,AQW,AC,*VD,*LD,*AC	整数

图 8-83　STI（ASCII 字符串转 16 位整数）

3）字符串指令包括：STR _ CPY（字符串复制）、STR _ LEN（字符串长度）、SSTR _ CPY（从字符串复制子字符串）、STR _ CAT（字符串连接）、STR _ FIND（在字符串内查找字符串）和 CHR _ FIND（在字符串中查找第一个字符）指令。其指令样式分别如图 8-84 ~ 图 8-89 所示。



输入/输出	操作数	数据类型
IN	VB,常量字符串,*VD,*LD,*AC	字符串
OUT	VB,*VD,*LD,*AC	字符串

图 8-84　STR _ CPY（字符串复制）



输入/输出	操作数	数据类型
IN	VB,常量字符串,*VD,*LD,*AC	字符串
OUT	VB,IB,QB,MB,SB,SMB,LB,AC,*VD,*LD,*AC	字节

图 8-85　STR _ LEN（字符串长度）

SSTR_CPY (EN, ENO, IN, OUT, INDX, N)	输入/输出	操作数	数据类型
	IN	VB,常量字符串,*VD,*LD,*AC	字符串
	INDX,N	VB,IB,QB,MB,SB,SMB,LB,AC 常量,*VD,*LD,*AC	字节
	OUT	VB,*VD,*LD,*AC	字符串

图 8-86　SSTR _ CPY（从字符串复制子字符串）

STR_CAT (EN, ENO, IN, OUT)	输入/输出	操作数	数据类型
	IN	VB,常量字符串,*VD,*LD,*AC	字符串
	OUT	VB,LB,*VD,*LD,*AC	字符串

图 8-87　STR _ CAT（字符串连接）

STR_FIND (EN, ENO, IN1, OUT, IN2)	输入/输出	操作数	数据类型
	IN1,IN2	VB,常量字符串,*VD,*LD,*AC	字符串
	OUT	VB,IB,QB,MB,SB,SMB,LB, AC,*VD,*LD,*AC	字节

图 8-88　STR _ FIND（在字符串内查找字符串）

CHR_FIND (EN, ENO, IN1, OUT, IN2)	输入/输出	操作数	数据类型
	IN1,IN2	VB,常量字符串,*VD,*LD,*AC	字符串
	OUT	VB,IB,QB,MB,SB,SMB,LB, AC,*VD,*LD,*AC	字节

图 8-89　CHR _ FIND（在字符串中查找第一个字符）

问 3：字符和字符串指令基本规律怎样?

答：当常量字符串被直接输入程序或数据块，该字符串必须用双引号（“字符串常量”）。

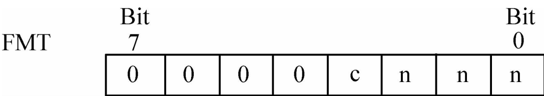
字符串的第一个字节定义字符串的长度，即字符数。字符串的长度可为 0～254 个字符，字符串最长为 255 个字节（长度字节 +254 个字符）。下面是字符串数据类型的格式。

长度	字符1	字符2	字符3	字符4	······	字符254
字节0	字节1	字节2	字节3	字节4	······	字节254

字符转换指令基本规律描述如下：

ITA 整数转换 ASCII 指令将 16 位整数（IN）转换成 ASCII 字符。格式（FMT）指定小数点右侧的位数，以及决定将小数点显示为逗号还是点号。转换结果置于从 OUT 开始的 8 个连续字节中。

以下是 ITA 指令的操作数（FMT）格式定义：



FMT

Bit 7							Bit 0
0	0	0	0	c	n	n	n

FMT 的高 4 位（Bit7 ~ Bit4）必须为 0，低 4 位（Bit3 ~ Bit0）分别是“cnnn”。其中：

“nnn”指定输出小数点右侧的位数，nnn 域的有效范围是 0 ~ 5，指定 nnn 域的数值为 0 时，输出不显示小数点。当指定 nnn 域大于 5 时，执行 ITA 指令将用 ASCII 的“空格键”符号填充满输出。

“c”位指定是使用逗号（c = 1）还是使用小数点（c = 0）作为整数和小数之间的分隔符。

根据下列规则对输出进行格式化：

①当 I 的数值为正整数时，写入输出缓冲器不带符号。

②当 I 的数值为负整数时，负值写入输出缓冲器，带起始负号（ - ）。

③小数点左侧的起首零（与小数点相邻的数字除外）被压缩。

④输出缓冲器中的数值右侧对齐。

ITA 指令实例程序如图 8-90 所示，调试状态见表 8-16。

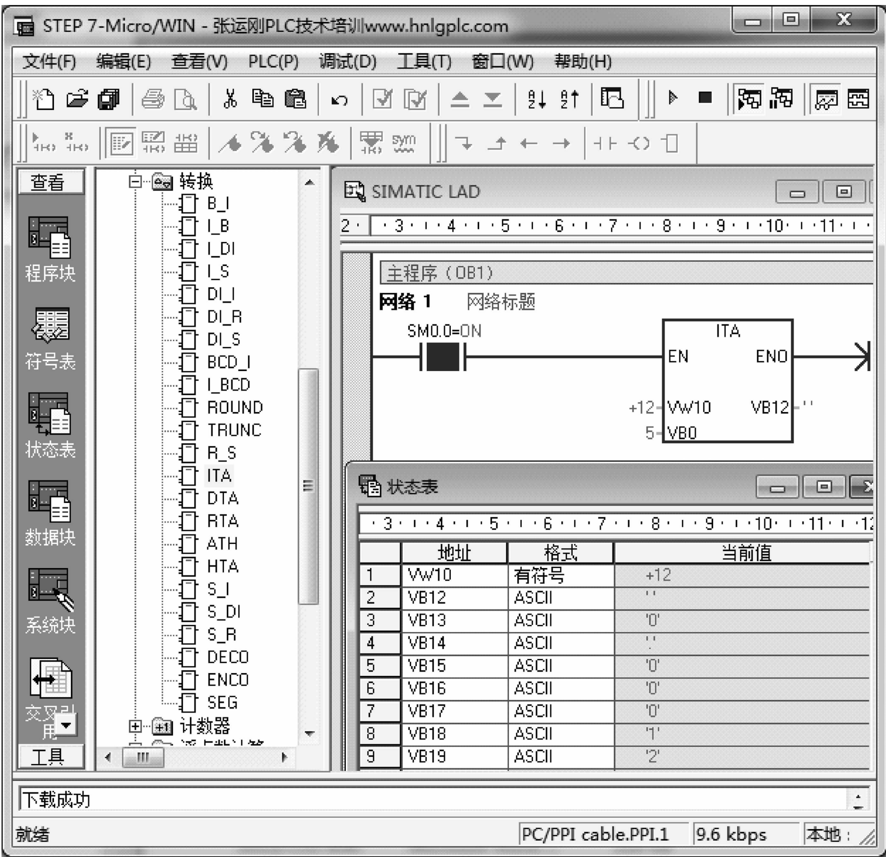


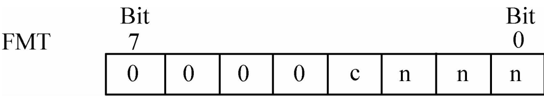
图 8-90　ITA 指令应用

表 8-16　ITA 指令调试状态表

FMT （VB0）	IN （VW10）	OUT							
		VB12	VB13	VB14	VB15	VB16	VB17	VB18	VB19
0 或 8	12 345	“ ”	“ ”	“ ”	“1”	“2”	“3”	“4”	“5”
6 及以上 （设 c＝0）	12 345	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
3	12 345	“ ”	“ ”	“1”	“2”	“.”	“3”	“4”	“5”
5	12	“ ”	“0”	“.”	“0”	“0”	“0”	“1”	“2”
3	－12	“ ”	“ ”	“－”	“0”	“.”	“0”	“1”	“2”

DTA 双整数转换 ASCII 指令将双字（IN）转换成 ASCII 字符。FMT 指定小数点的位数和现实格式。转换结果放置于从 OUT 开始的 12 个连续字节中。

以下是 DTA 指令的操作数 FMT 格式定义：



FMT 的高 4 位（Bit7～Bit4）必须为 0，低 4 位（Bit3～Bit0）分别是“cnnn”。其中：

“nnn”指定输出缓冲器中小数点右侧的位数，nnn 域的有效范围是 0～5。当 nnn 域的数值为 0 时，输出缓冲器不显示小数点，当 nnn 域大于 5 时，执行 DTA 指令时将用 ASCII 码的“空格键”符号填充满输出缓冲器。

“c”位指定是使用逗号（c＝1）还是使用小数点（c＝0）作为整数和小数之间的分隔符。

根据下列规则对输出缓冲器进行格式化：

①正值时写入输出缓冲器，不带符号。

②负值时写入输出缓冲器，带起始负号（－）。

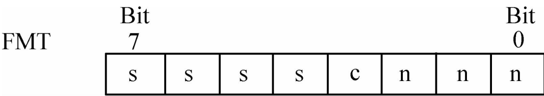
③小数点左侧的起首零（与小数点相邻的数字除外）被压缩。

④输出缓冲器中的数值右对齐。

DTA 指令实例程序如图 8-91 所示，调试状态见表 8-17。

RTA 实数转换 ASCII 指令将实数（IN）数值转换成 ASCII 字符。格式（FMT）指定小数点的位数，以及决定将小数点表示为逗号或点号及输出缓冲器占用的长度。转换结果置于从 OUT 开始的输出缓冲器中。结果 ASCII 字符的数目（或长度）范围为 3～15 个字符。

以下是 RTA 指令的格式操作数（FMT）定义：



“ssss”指定 OUT 输出缓冲器的大小，ssss 域指定 3～15 有效，指定 0～2 无效。

“nnn”指定输出缓冲器中小数点位数，nnn 域的有效范围是 0～5。将小数点右面的位数指定为 0 会使数值显示为不带小数点。当 nnn 数值大于 5 时或当指定的输出字符串长度太小（0～2）无法存储转换的数值时，输出缓冲器将用 ASCII 码的“空格键”字符填满。

“c”位指定是使用逗号（c ＝ 1）还是使用小数点（c ＝ 0）作为整数和小数之间的分隔符。

根据下列规则对输出缓冲器进行格式化：

①正数写入输出缓冲器，不带符号。
②负数写入输出缓冲器，带起始负号（－）。
③小数点左侧的起首零（与小数点相邻的数字除外）被压缩。
④小数点右侧的数值进位，使之符合小数点右侧指定的位数。
⑤输出缓冲器的位数必须最少比小数点右侧位数多 3 字节。
⑥输出缓冲器中的数值右对齐。

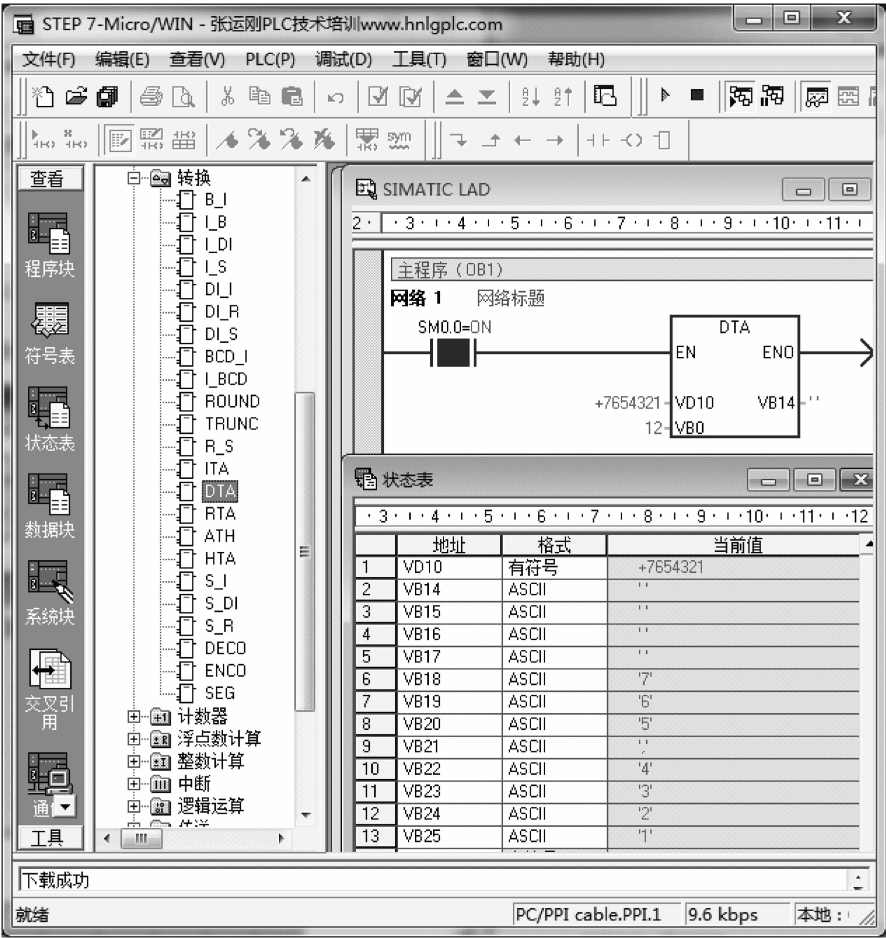


图 8-91　DTA 指令应用

表 8-17　DTA 指令调试状态表

FMT	IN	OUT											
（VB0）	（VD10）	VB14	VB15	VB16	VB17	VB18	VB19	VB20	VB21	VB22	VB23	VB24	VB25
0 或 8	7 654 321	“ ”	“ ”	“ ”	“ ”	“ ”	“7”	“6”	“5”	“4”	“3”	“2”	“1”
14 及以上（c＝1）	7 654 321	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
12	7 654 321	“ ”	“ ”	“ ”	“ ”	“7”	“6”	“5”	“,”	“4”	“3”	“2”	“1”
12	－7 654 321	“ ”	“ ”	“ ”	“－”	“7”	“6”	“5”	“,”	“4”	“3”	“2”	“1”
12	21	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“0”	“,”	“0”	“0”	“2”	“1”

RTA 指令实例程序如图 8-92 所示，调试状态见表 8-18。

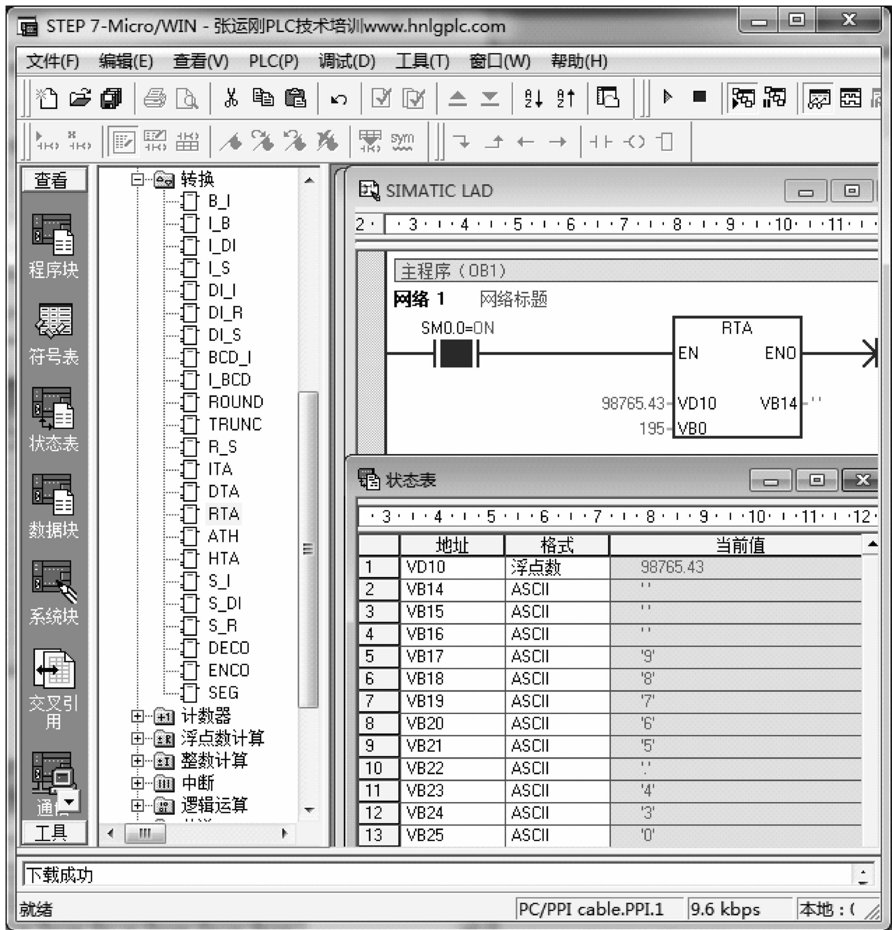


图 8-92　RTA 指令应用

表 8-18　RTA 指令调试状态表

FMT（VB0）	IN（VD10）	OUT											
		VB14	VB15	VB16	VB17	VB18	VB19	VB20	VB21	VB22	VB23	VB24	VB25
16#C0	12 346. 789	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“1”	“2”	“3”	“4”	“7”
16#0	12 346. 789	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
16#C2	12 346. 789	“ ”	“ ”	“ ”	“ ”	“ ”	“1”	“2”	“3”	“4”	“.”	“7”	“9”
16#C3	－12 346. 789	“ ”	“ ”	“ ”	“－”	“1”	“2”	“3”	“4”	“.”	“7”	“8”	“9”
16#CB	0. 01	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“0”	“,”	“0”	“1”	“0”

HTA 十六进制数转换 ASCII 指令将从输入字节（IN）开始的十六进制数字转换成从 OUT 开始的 ASCII 字符。参数（LEN）指定欲转换的十六进制数字位数的长度。可转换的最大十六进制数字位数长度为 255。

HTA 指令实例程序如图 8-93 所示。

ATH ASCII 转换十六进制数指令将从 IN 开始的 ASCII 字符号码（LEN）转换成从 OUT 开始的十六进制数字。ASCII 字符串的最大长度为 255 个字符。

ATH 指令应用如图 8-94 所示。

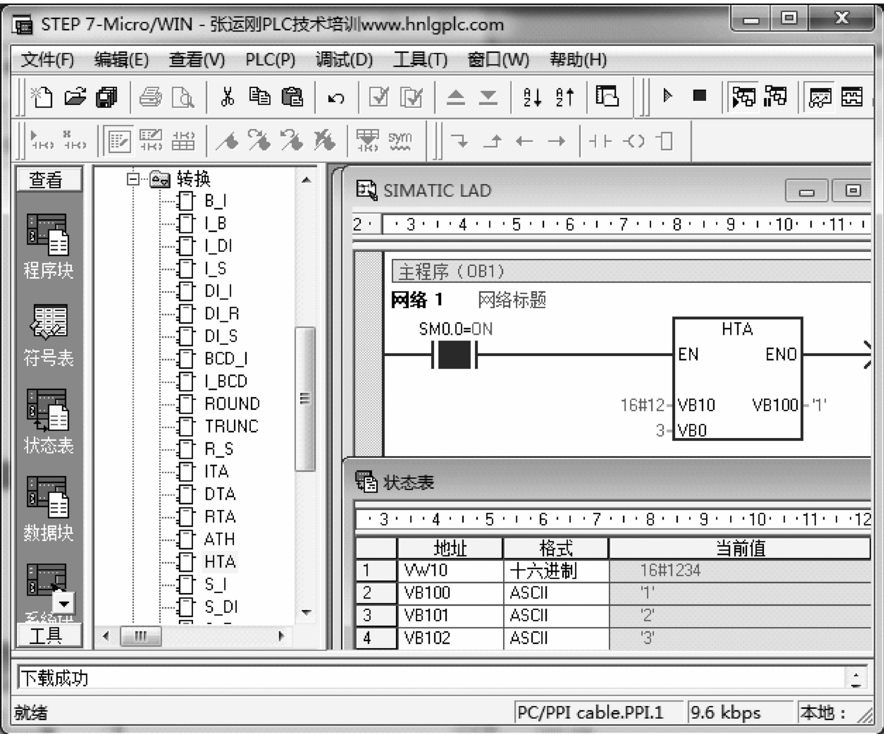


图 8-93　HTA 指令应用

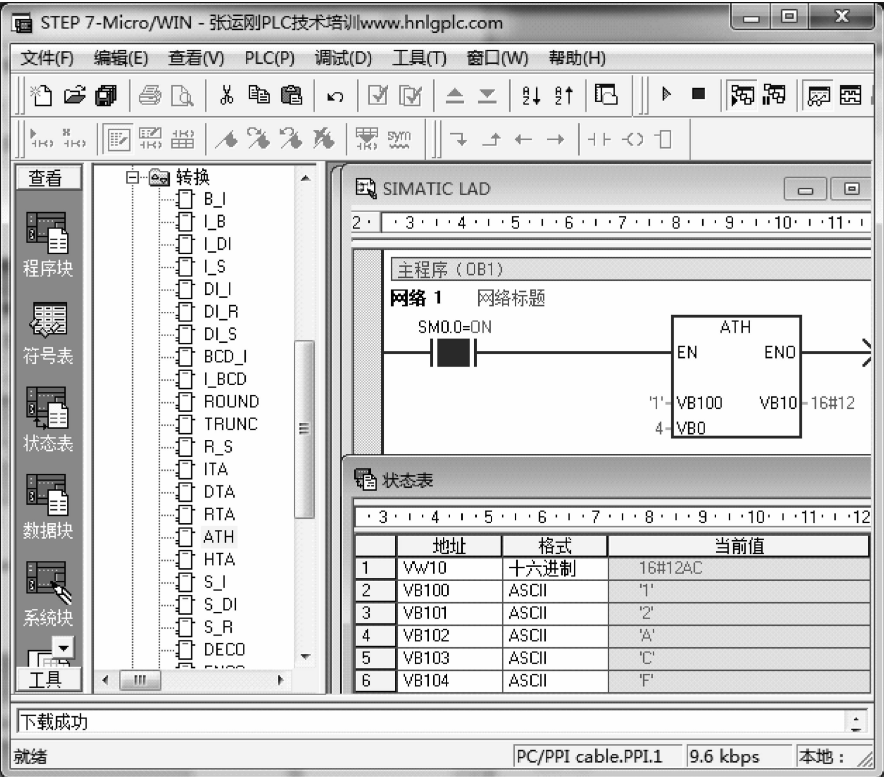


图 8-94　ATH 指令应用

HTA 和 ATH 有效的 ASCII 字符见表 8-19，限制于十六进制数值。

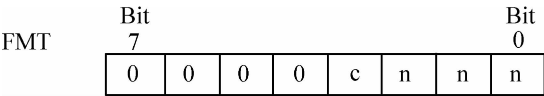
表 8-19　HTA 和 ATH 指令有效 ASCII 字符表

数字 ASCII	0	1	2	3	4	5	6	7	8	9
十六进制值	30	31	32	33	34	35	36	37	38	39
字母 ASCII	A	B	C	D	E	F				
十六进制值	41	42	43	44	45	46				

字符串转换指令基本规律描述如下：

ITS 将整数转换为字符串指令将 16 位整数（IN）转换为长度为 8 个字符的 ASCII 字符串。格式（FMT）指定小数点的位数，小数点显示为逗号或者是点。结果字符串写入从 OUT 开始的 9 个连续字节中。

ITS 的（FMT）操作数格式定义：



c = 逗号（1）或小数点（0）。c 位指定是使用逗号（c = 1）还是使用小数点（c = 0）作为整数和小数之间的分隔符。

nnn = 小数点的位数。输出缓冲器中小数点的位数由 nnn 域指定，nnn 域的有效范围是 0 ~5。将小数点右面的位数指定为 0 会使数值显示为不带小数点。当 nnn 数值大于 5 时，输出显示为 8 个 ASCII“空格键”字符的字符串。

FMT 的高 4 位（Bit7 ~ Bit4）必须为零。

输出缓冲器占用 9 个字节，输出字符串的长度始终为 8 个字符。输出缓冲器是 ASCII 常量字符串数据类型的格式：字符串是一系列字符，每个字符使用一个字节存储。字符串的第一个字节定义字符串的长度，即字符数，第二个字节至第九个字节是 ASCII 字符。

输出字符串的规律如下：

①整数不带符号写入输出缓冲器。

②负数带起始负号（ - ）写入输出缓冲器。

③小数点左面的起始零（除紧靠小数点的数字外）被压缩。

④小数点右面的数值被进位，使之符合指定的小数点的位数。

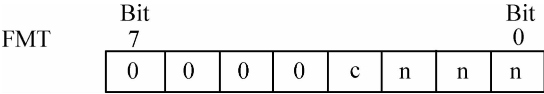
⑤输出字符串的大小，最小必须比小数点右面的位数大 3 字节。

⑥输出字符串中的数值必须右对齐。

ITS 指令实例程序如图 8-95 所示，调试状态见表 8-20。

DTS 将双整数转换为字符串指令将双整数 IN 转换为长度为 12 个字符的 ASCII 字符串。格式（FMT）指定小数点的位数，小数点显示为逗号或者点。结果字符串写入从 OUT 开始的 13 个连续字节中。

DTS 的格式（FMT）操作数定义：



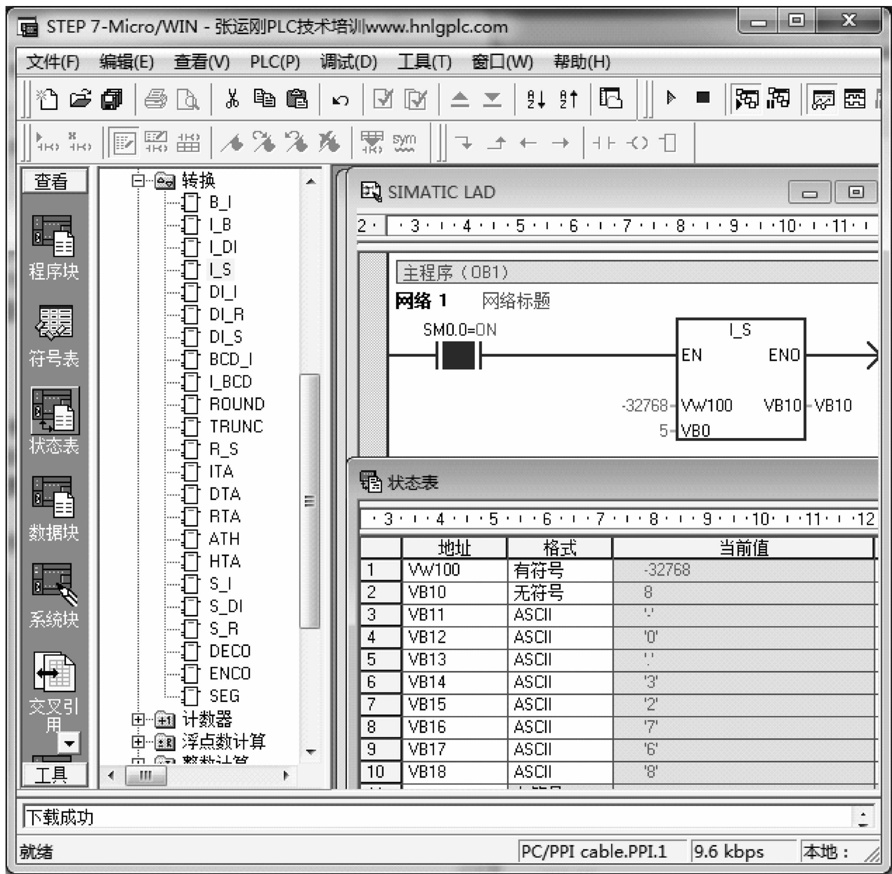


图 8-95　ITS 指令实例程序

表 8-20　ITS 指令调试状态表

FMT（VB0）	IN（VW10）	OUT								
		VB50	VB51	VB52	VB53	VB54	VB55	VB56	VB57	VB58
2	1 234	8	“ ”	“ ”	“ ”	“1”	“2”	“.”	“3”	“4”
2	－1 234	8	“ ”	“ ”	“－”	“1”	“2”	“.”	“3”	“4”
2	－1	8	“ ”	“ ”	“ ”	“－”	“0”	“.”	“0”	“1”
10	1 234	8	“ ”	“ ”	“ ”	“1”	“2”	“，”	“3”	“4”

c＝逗号（1）或小数点（0）。c 位指定是使用逗号（c＝1）还是使用小数点（c＝0）作为整数和小数之间的分隔符。

nnn＝小数点的位数。输出缓冲器中小数点的位数由 nnn 域指定，nnn 域的有效范围是 0～5。将小数点右面的位数指定为 0 会使数值显示为不带小数点。当 nnn 数值大于 5 时，输出显示为 12 个 ASCII“空格键”字符的字符串。

FMT 的高 4 位（Bit7 ~ Bit4）必须为零。

输出缓冲器占用 13 字节，输出字符串的长度始终为 12 字符。输出缓冲器是 ASCII 常量字符串数据类型的格式：字符串是一系列字符，每个字符作为一个字节存储。字符串的第一个字节定义字符串的长度，即字符数，第二个字节至第十三个字节是 ASCII 字符。

输出字符串的规律如下：

①整数不带符号写入输出缓冲器。

②负数带起始负号（－）写入输出缓冲器。

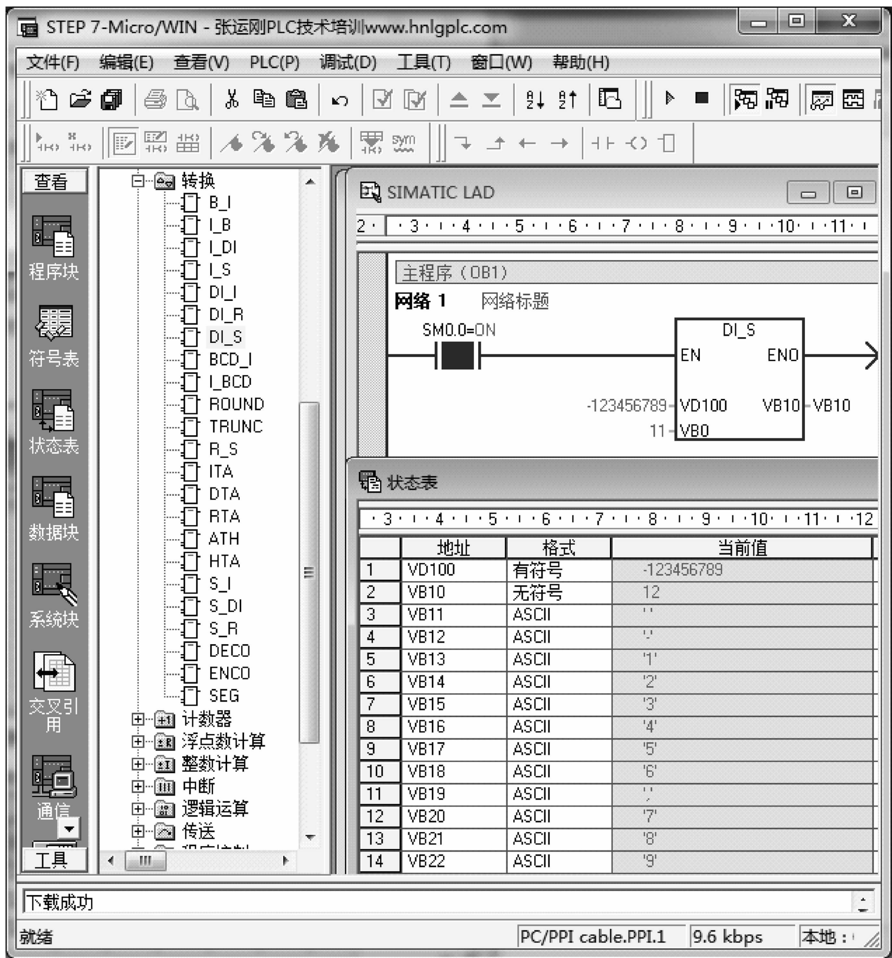
③小数点左面的起始零（除紧靠小数点的数字外）被压缩。

④小数点右面的数值被进位，使之符合指定的小数点的位数。

⑤输出字符串的大小最小必须比小数点右面的位数大 3 个字节。

⑥输出字符串中的数值必须右对齐。

DTS 指令实例程序如图 8-96 所示，调试状态见表 8-21。



	地址	格式	当前值
1	VD100	有符号	-123456789
2	VB10	无符号	12
3	VB11	ASCII	' '
4	VB12	ASCII	'-'
5	VB13	ASCII	'1'
6	VB14	ASCII	'2'
7	VB15	ASCII	'3'
8	VB16	ASCII	'4'
9	VB17	ASCII	'5'
10	VB18	ASCII	'6'
11	VB19	ASCII	'.'
12	VB20	ASCII	'7'
13	VB21	ASCII	'8'
14	VB22	ASCII	'9'

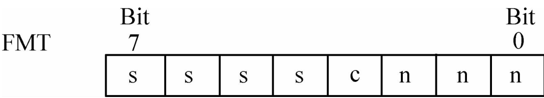
图 8-96　DTS 指令实例程序

表 8-21　DTS 指令调试状态表

FMT	IN	OUT											
(VB0)	(VD10)	VB51	VB52	VB53	VB54	VB55	VB56	VB57	VB58	VB59	VB60	VB61	VB62
16#4	123 498 765	“ ”	“ ”	“1”	“2”	“3”	“4”	“9”	“.”	“8”	“7”	“6”	“5”
16#4	-123 498 765	“ ”	“-”	“1”	“2”	“3”	“4”	“9”	“.”	“8”	“7”	“6”	“5”
16#6	-123 498 765	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
16#4	123	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“0”	“.”	“0”	“1”	“2”	“3”
16#4	-1 123 498 765	“-”	“1”	“1”	“2”	“3”	“4”	“9”	“.”	“8”	“7”	“6”	“5”

RTS 将实数转换为字符串指令将实数值（IN）转换为 ASCII 字符串。格式（FMT）指定小数点的转换精度，无论小数点显示为逗号还是句点，亦无论输出字符串的长度是多少。转换结果放置在以 OUT 开始的字符串中。结果字符串长度在格式（FMT）中指定，只能是 3～15 个字符。S7-200 使用的实数格式最多可支持 7 个高位数字。如果显示 7 个以上高位数字会产生进位错误。

RTS 的格式（FMT）操作数定义：



ssss = 输出字符串长度，3～15 有效，ssss = 0、1 或 2 无效。

c = 逗号（1）或小数点（0），c 位指定是使用逗号（c = 1）还是使用小数点（c = 0）作为整数和小数之间的分隔符。

nnn = 小数点的位数，nnn 域的有效范围是 0～5。将小数点的位数指定为 0 会使数值显示为不带小数点。当 nnn 数值大于 5 时，输出显示为 8 个 ASCII“空格键”字符的字符串。

输出字符串的规律如下：

①整数不带符号写入输出缓冲器。

②负数带起始负号（-）写入输出缓冲器。

③小数点左面的起始零（除紧靠小数点的数字外）被压缩。

④小数点右面的数值被进位，使之符合指定的小数点右面的位数。

⑤输出字符串的大小，最小必须比小数点右面的位数大 3 字节。

⑥输出字符串中的数值必须右对齐。

RTS 指令实例程序如图 8-97 所示，调试状态见表 8-22。

表 8-22　RTS 指令程序调试状态表

FMT	IN	OUT								
(VB0)	(VD10)	VB50	VB51	VB52	VB53	VB54	VB55	VB56	VB57	VB58
16#04	12 349.876 5	8	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
16#84	-1 234.987 65	8	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”
16#84	123.498 76	8	“1”	“2”	“3”	“.”	“4”	“9”	“8”	“8”
16#84	-23.0	8	“-”	“2”	“3”	“.”	“0”	“0”	“0”	“0”
16#84	-11 234.0	8	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”	“ ”

STR 将子字符串转换为实数指令将 ASCII 字符串（IN）转换为实数值。INDX 值通常设为 1，从字符串的第一个字符开始转换，允许将 INDX 值设为其他值，在字符串中的不同点开始转换。

STR 指令实例程序如图 8-98 所示。

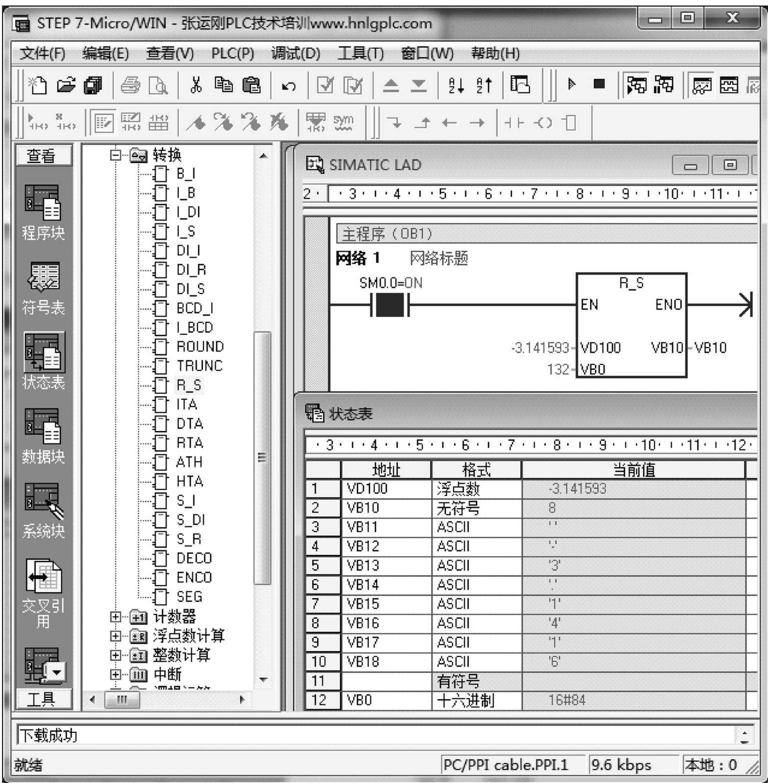


图 8-97　RTS 指令实例程序

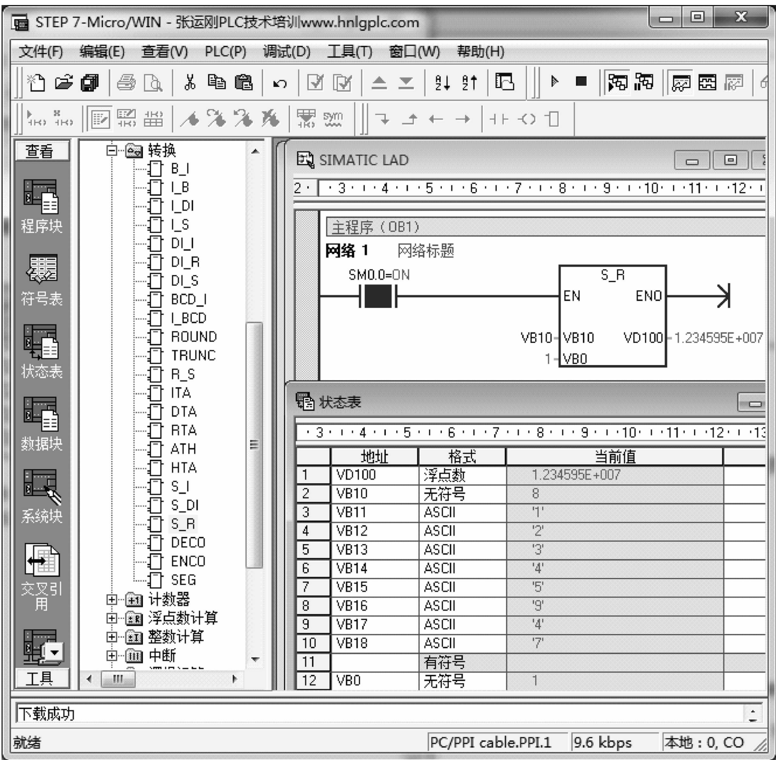


图 8-98　STR 指令应用

STD 将子字符串转换为双整数指令将 ASCII 字符串（IN）转换为双整数值。INDX 值通常设为 1，从字符串的第一个字符开始转换，允许将 INDX 值设为其他值，在字符串中的不同点开始转换。

STD 指令实例程序如图 8-99 所示。

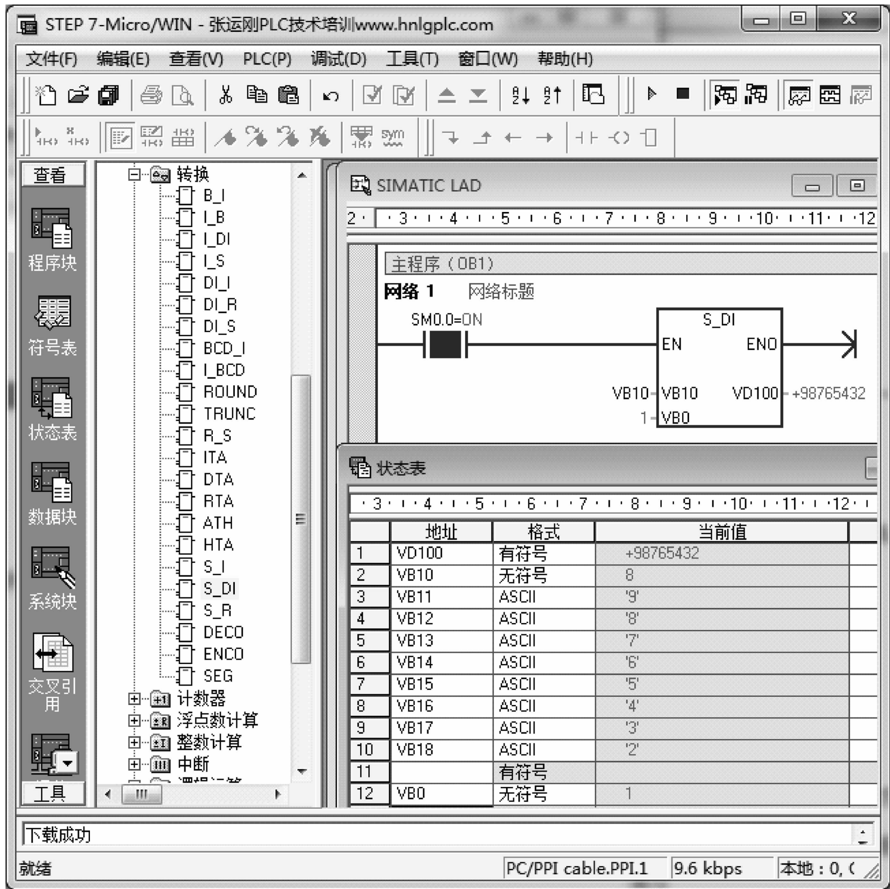


图 8-99　STD 指令应用

STI 将子字符串转换为整数指令将 ASCII 字符串（IN）转换为 16 位整数值。INDX 值通常设为 1，从字符串的第一个字符开始转换，允许将 INDX 值设为其他值，在字符串中的不同点开始转换。

STI 指令实例程序如图 8-100 所示。

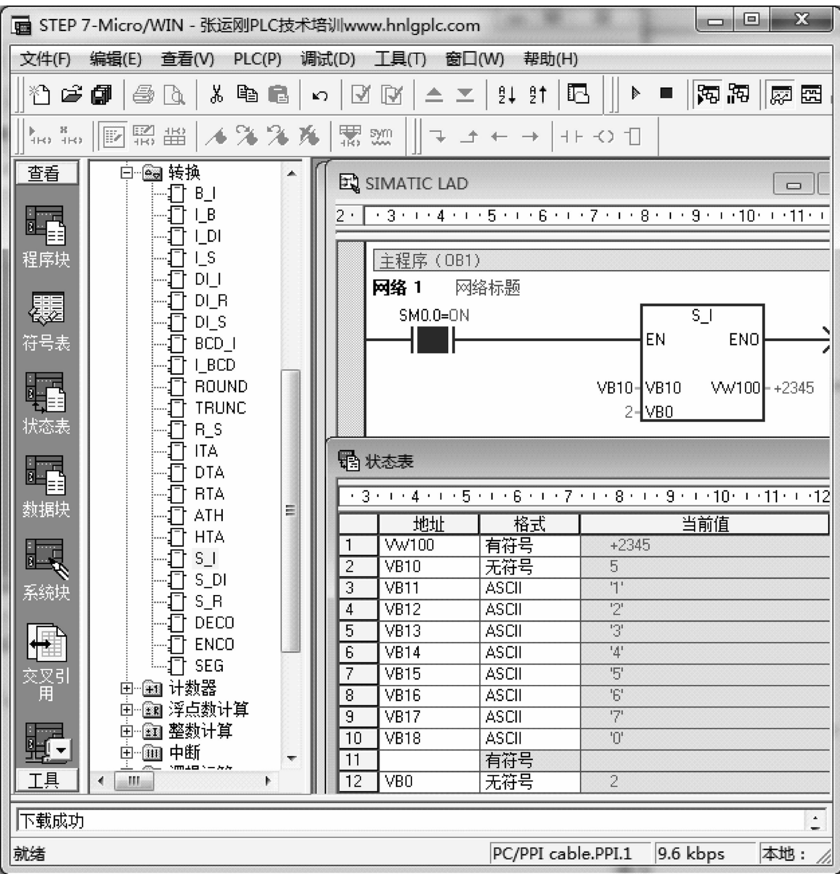


图 8-100　STI 指令应用

字符串指令基本规律描述如下：

STR _ CPY 字符串复制指令是将 IN 指定的字符串复制到 OUT 指定的字符串。常量字符串参数最长为 126 个字节。

STR _ CPY 字符串复制指令如图 8-101 所示。

STR _ LEN 计算字符串长度指令是计算 IN 指定的字符串长度数值传送到 OUT 指定的字节中，常量字符串参数最长为 126 个字节。计算字符串长度指令如图 8-102 所示。

SSTR _ CPY 子字符串复制指令是把 IN 指定的字符串中从第 INDX 指定的字符开始连续 N 指定的字符个数复制到 OUT 指定的字符串中。常量字符串参数最长为 126 个字节。

SSTR _ CPY 子字符串复制指令如图 8-103 所示。

STR _ CAT 连接字符串指令将 IN 指定的字符串连接至 OUT 指定的字符串之后，使 OUT 指定的字符串增加了 IN 指定的字符数。常量字符串参数最长为 126 个字节。

STR _ CAT 连接字符串指令如图 8-104 所示，执行指令后状态图如图 8-105 所示。

STR _ FIND 字符串内查找字符串指令，在字符串 IN1 中搜索第一次出现的字符串 IN2。搜索从 IN1 第一个位置开始，如果找到一个与字符串 IN2 完全符合的字符系列，该系列的第一个字符位置被写入 OUT。如果在字符串 IN1 中未找到字符串 IN2，OUT 被设置为 0。单个常量字符串最长为 126 个字节，两个常量字符串综合最长为 240 个字节。

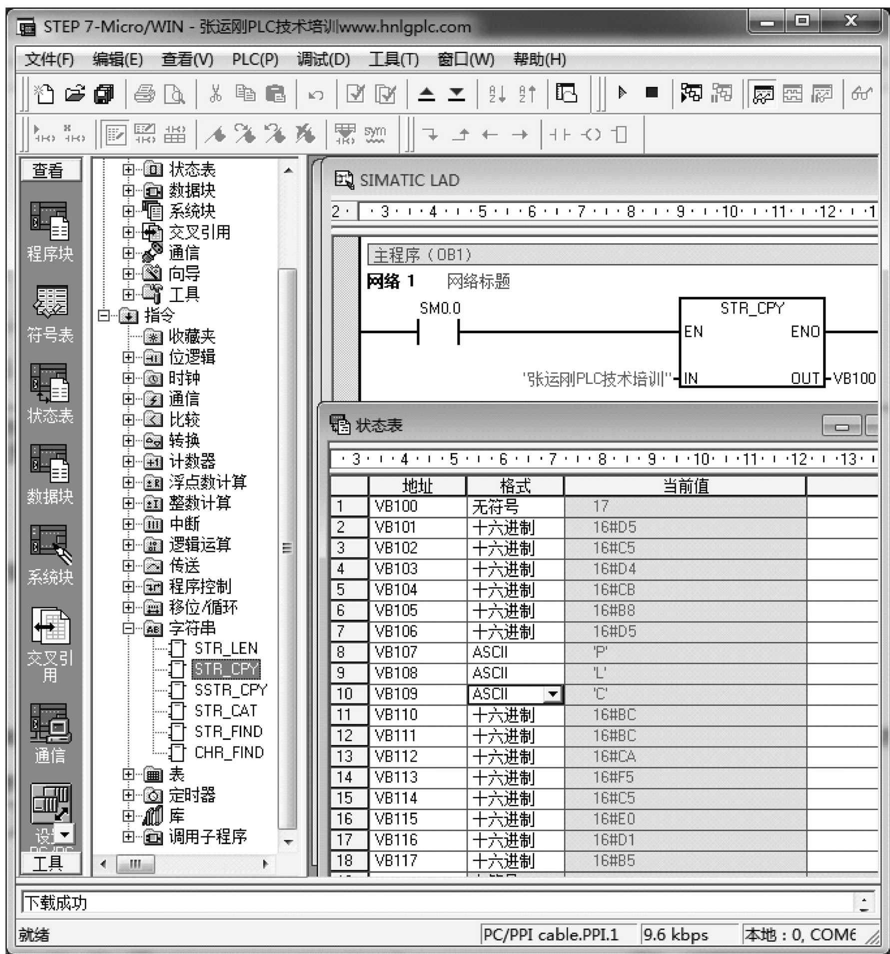


图 8-101　STR _ CPY 字符串复制指令

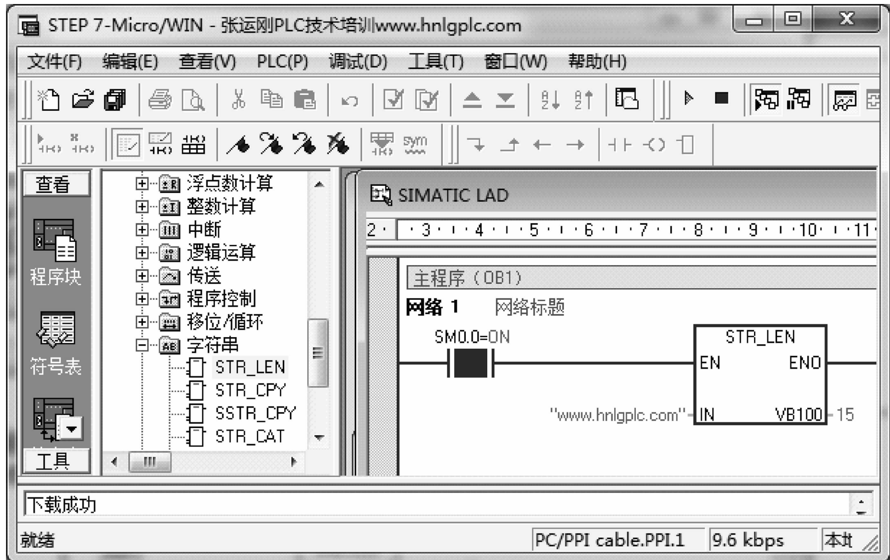


图 8-102　STR _ LEN 计算字符串长度指令

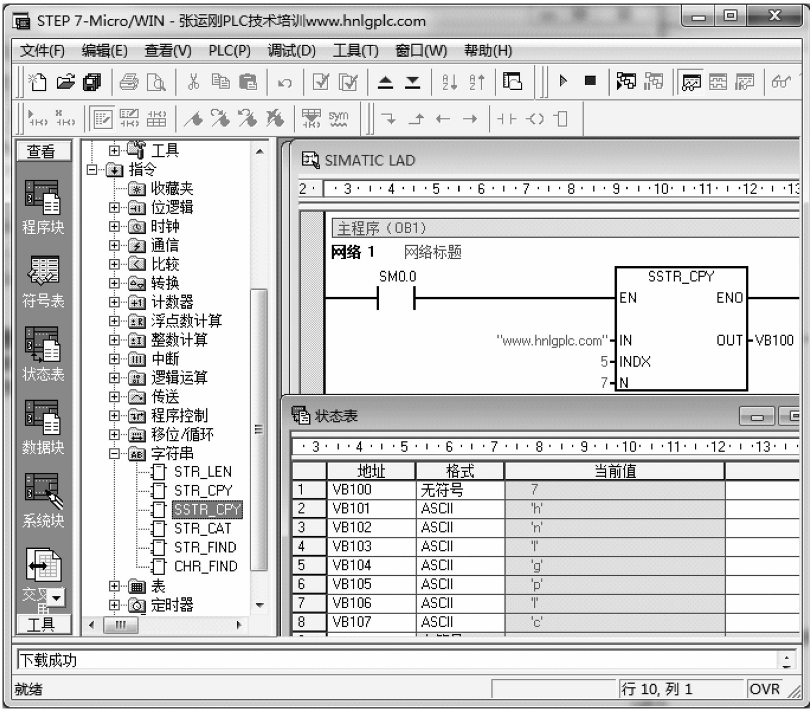


图 8-103 SSTR _ CPY 子字符串复制指令

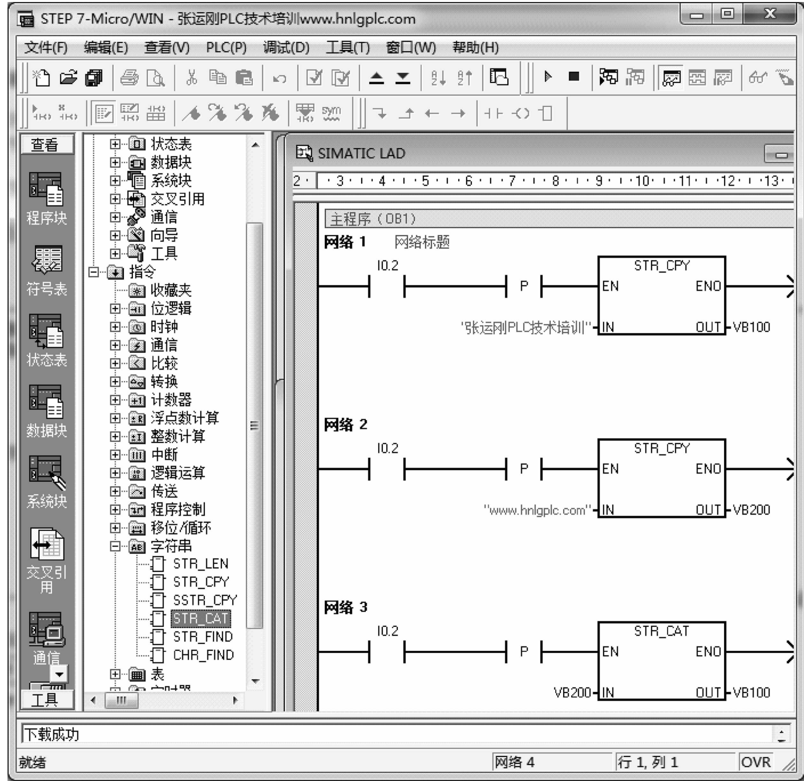


图 8-104 STR _ CAT 连接字符串指令

	地址	格式	当前值
1	VB100	无符号	32
2	VB101	十六进制	16#D5
3	VB102	十六进制	16#C5
4	VB103	十六进制	16#D4
5	VB104	十六进制	16#CB
6	VB105	十六进制	16#B8
7	VB106	十六进制	16#D5
8	VB107	ASCII	'P'
9	VB108	ASCII	'L'
10	VB109	ASCII	'C'
11	VB110	十六进制	16#BC
12	VB111	十六进制	16#BC
13	VB112	十六进制	16#CA
14	VB113	十六进制	16#F5
15	VB114	十六进制	16#C5
16	VB115	十六进制	16#E0
17	VB116	十六进制	16#D1
18	VB117	十六进制	16#B5
19	VB118	ASCII	'w'
20	VB119	ASCII	'w'
21	VB120	ASCII	'w'
22	VB121	ASCII	'.'
23	VB122	ASCII	'h'
24	VB123	ASCII	'n'
25	VB124	ASCII	'l'
26	VB125	ASCII	'g'
27	VB126	ASCII	'p'
28	VB127	ASCII	'l'
29	VB128	ASCII	'c'
30	VB129	ASCII	'.'
31	VB130	ASCII	'c'
32	VB131	ASCII	'o'
33	VB132	ASCII	'm'

图 8-105　STR _ CAT 状态图

STR _ FIND 字符串内查找字符串指令如图 8-106 所示。

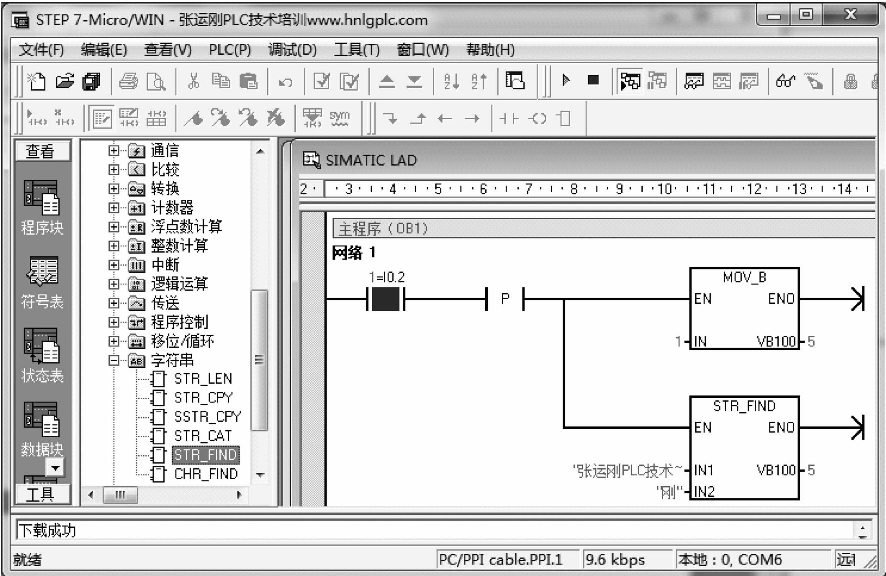


图 8-106　STR _ FIND 字符串内查找字符串指令

CHR _ FIND 在字符串中查找第一个字符指令，在字符串 IN1 中搜索与字符串 IN2 中的字符集中的任何相符的字符，查找从 IN1 中第一个位置开始，如果找到一个相符的字符，该字符的位置写入 OUT；如果没有找到相符的字符，OUT 被设置为 0。单个常量字符串最长为 126 个字节，两个常量字符串综合最长为 240 个字节。CHR _ FIND 指令程序如图 8-107 所示。

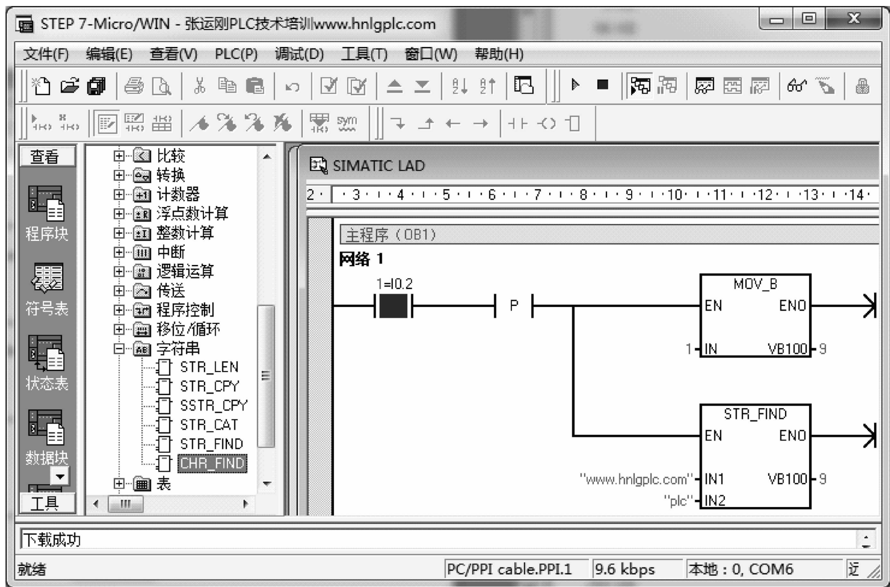


图 8-107　CHR _ FIND 指令

问 4：在应用字符和字符串指令时需要注意些什么？

答：在应用字符和字符串指令时需要注意以下几点：

一是间接寻址时注意不要超出寻址范围。

二是注意结果超出操作数范围，特别是使用 ATH 指令时需要特别注意。

三是当执行 ITA 等指令时，注意当指定 nnn 域大于 5 时，将没有正确结果输出。

四是当执行 RTA 等指令时，注意“ssss”指定为 0 ~ 2 无效，只能在 3 ~ 15 的范围内。

五是注意脉冲执行和连续执行是有区别的。

8.6　编码/译码转换

问：什么时候会使用到编码解码指令？
编码解码指令样式是怎样的？
编码解码指令基本规律怎样？
在应用编码解码指令时需要注意些什么？

问 1：什么时候会使用到编码解码指令？

答：当遇到一个位对应一个数值的逻辑，或者一个数值对应一个位逻辑时，可以使用编码或者解码指令。当然还需要一些附加条件，就是这些位需要连续的位，这些数值也需要连续的数值。

问 2：编码解码指令样式是怎样的？

答：编码和解码指令样式如图 8-108 和图 8-109 所示。



输入/输出	操作数	数据类型
IN	VW,IW,QW,MW,SMW,LW,SW,AIW,T,C,AC,常量,*VD,*AC,*LD	字
OUT	VB,IB,QB,MB,SMB,LB,SB,AC,*VD,*LD,*AC	字节

图 8-108　编码指令样式



输入/输出	操作数	数据类型
IN	VB,IB,QB,MB,SMB,LB,SB,AC,常量,*VD,*LD,*AC	字节
OUT	VW,IW,QW,MW,SMW,LW,SW,AQW,T,C,AC,*VD,*AC,*LD	字

图 8-109　解码指令样式

问 3：编码解码指令基本规律怎样？

答：编码/解码指令基本规律是：

ENCO 编码指令将输入字（IN）中把最低接通位的位号写入输出字节（OUT）的最低"半字节"（低 4 位）中。

ENCO 指令应用如图 8-110 所示的程序中，VW0【共 2^{15} 位中第（2^m）位】中接通的位的排行第 m 位编译成数值 m 传送到 VB2 中，输入与输出详细关系见表 8-23，表中如果有多位接通，以最低位为准。

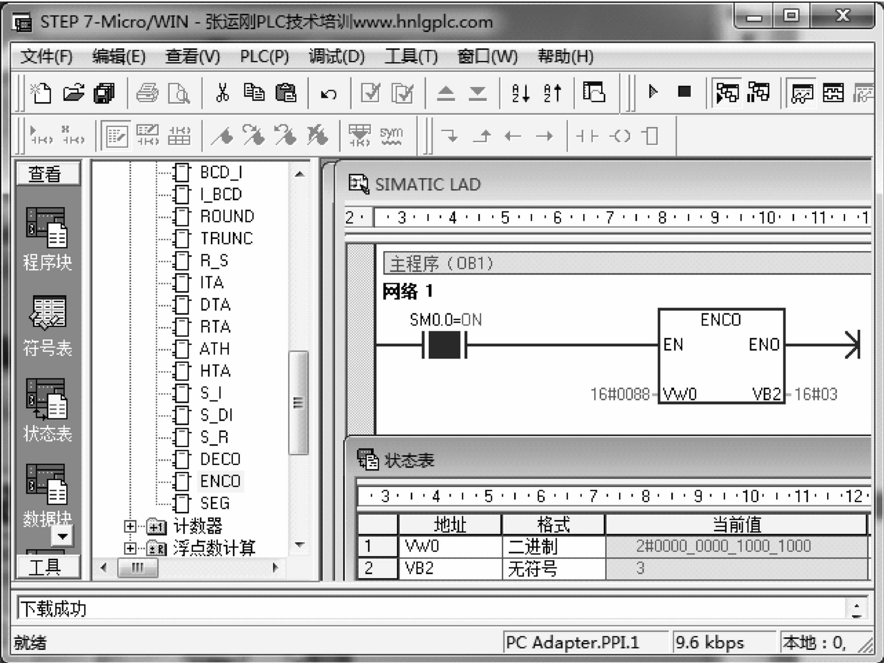


图 8-110　ENCO 指令实例程序

表 8-23 ENCO 动作示意

IN	状态	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0
	VW0	V0.7	V0.6	V0.5	V0.4	V0.3	V0.2	V0.1	V0.0	V1.7	V1.6	V1.5	V1.4	V1.3	V1.2	V1.1	V1.0
	第 m 位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
OUT	VB2	VB2 = m(本例 m = 3)															

ENCO 编码指令的动作可以理解为源中的一个位对应目标中的数（即：位→数），详细解析见表 8-24。表 8-24 中的意思是：

当 V1.0 = 1，VB2 = 0；

当 V1.1 = 1，VB2 = 1；

当 V1.2 = 1，VB2 = 2；

当 V1.3 = 1，VB2 = 3；

……

当 V0.7 = 1，VB2 = 15。

电梯轿厢所在楼层显示使用 ENCO 编码指令很容易实现，I1.0 ~ I1.7 相当于第一层 ~ 第八层的楼层检测、I0.0 ~ I0.7 相当于第九层 ~ 第十六层的楼层检测开关，轿厢所在楼层的位置数值在 VB0 里，示意图如图 8-111 所示，程序如图 8-112 所示。

表 8-24 编码指令解释

OUT VW0(假设只有一位为 1)		OUT VB2 数值（0 ~ 15）
对 应 位	状 态	
V0.7	1	15
V0.6	1	14
V0.5	1	13
V0.4	1	12
V0.3	1	11
V0.2	1	10
V0.1	1	9
V0.0	1	8
V1.7	1	7
V1.6	1	6
V1.5	1	5
V1.4	1	4
V1.3	1	3
V1.2	1	2
V1.1	1	1
V1.0	1	0

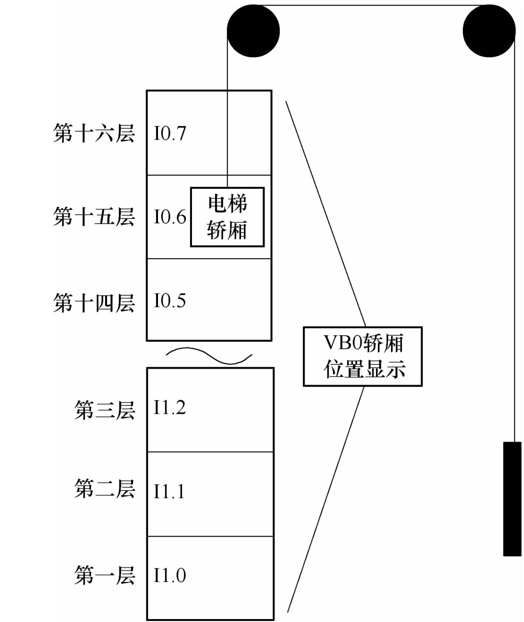


图 8-111 电梯轿厢所在楼层显示

DECO 解码指令输入字节（IN）最低“半字节”（4 位）表示的数相对应设置输出字（OUT）中的位，输出字的其他所有位均设为 0。

在图 8-113 所示的程序中，VB10 二进制位低 4 位（V10.3、V10.2、V10.1、V10.0）的数（0 ~ 15）译码对应输出 VW12 中 V11.7、V11.6、V11.5、V11.4、V11.3、V11.2、V11.1、V11.0、V12.7、V12.6、V12.5、V12.4、V12.3、V12.2、V12.1、V12.0 的位，示意图如图 8-114 所示。

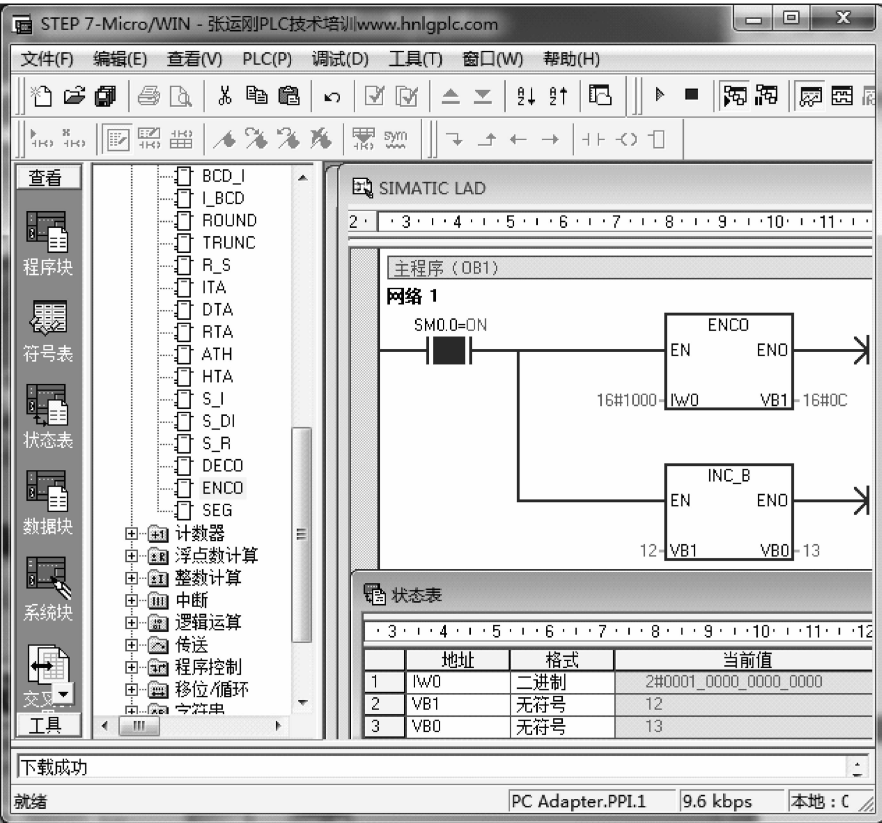


图 8-112　电梯轿厢所在楼层显示程序

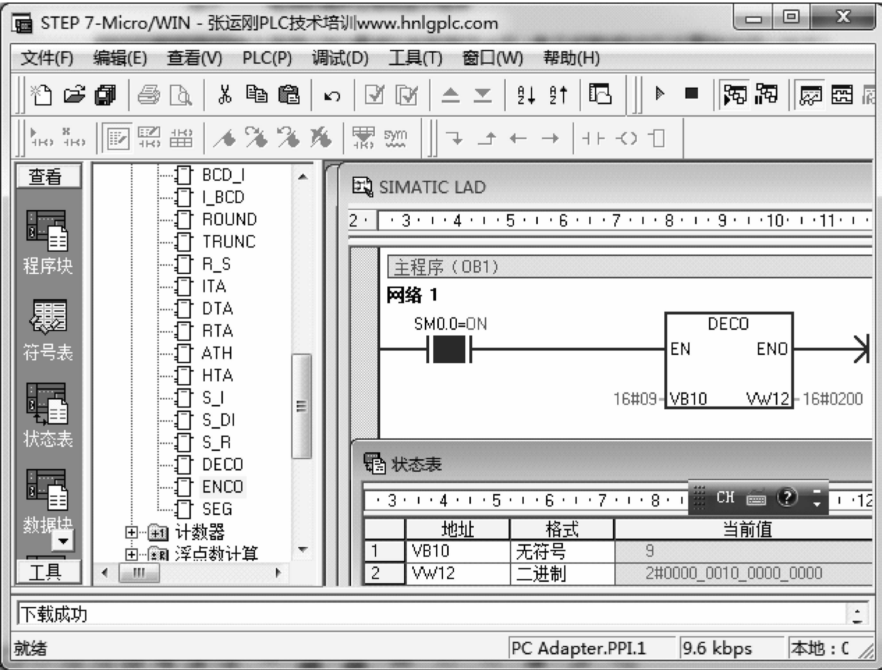


图 8-113　DECO 指令实例程序

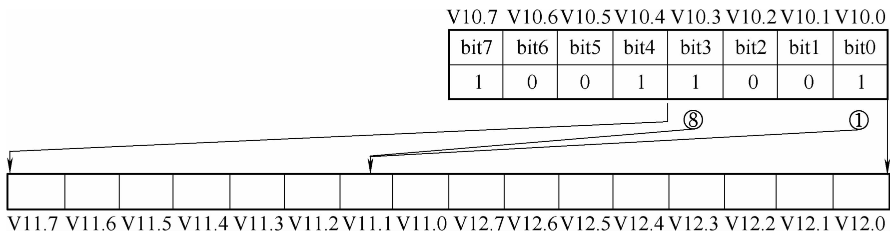


图 8-114　DECO 示意图

DECO 解码指令的动作可以理解为 IN 源中的数值对应 OUT 目标中的位（即：数→位），见表 8-25。表 8-25 中的具体意思是：

当 VB10 = 0，V12. 0 = 1；

当 VB10 = 1，V12. 1 = 1；

当 VB10 = 2，V12. 2 = 1；

当 VB10 = 3，V12. 3 = 1；

……

当 VB10 = 15，V11. 7 = 1。

表 8-25　DECO 解码指令表

IN VB10 数值(0 ~ 15)	OUT VW12	
	对　应　位	状　　态
15	V11. 7	1
14	V11. 6	1
13	V11. 5	1
12	V11. 4	1
11	V11. 3	1
10	V11. 2	1
9	V11. 1	1
8	V11. 0	1
7	V12. 7	1
6	V12. 6	1
5	V12. 5	1
4	V12. 4	1
3	V12. 3	1
2	V12. 2	1
1	V12. 1	1
0	V12. 0	1

DECO 解码的动作与电话自动交换机解码动作相似，IN 中的数（VB10）相当于拨进来的号码（在容量范围内），交换机根据号码自动接通对应的一部电话 VX. Y，示意图如图 8-115 所示。

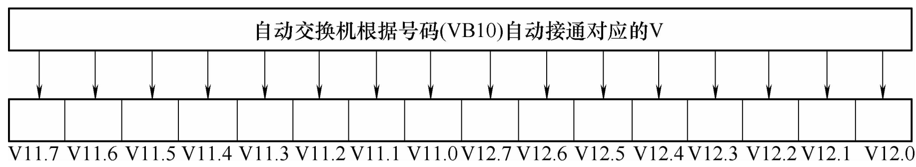


图 8-115　电话自动交换机示意图

问 4：在应用编码解码指令时需要注意些什么？

答：使用 ENCO 编码和 DECO 解码指令时需要注意以下事项：

一是间接寻址时注意不要超出寻址范围。

二是注意脉冲执行和连续执行是有区别的。

三是 ENCO 编码指令时，如果有多位接通，以最低位为准，其他位成为无效位。

第 9 章　加一减一逻辑指令

9.1　INC _ B/W/DW

> 问：INC 指令基本功能是什么?
> INC 指令样式是怎样的?
> 如何应用 INC 指令?
> 在应用 INC 指令时需要注意些什么?

问 1：INC 指令基本功能是什么?

答：执行 INC 一次，IN 中的变量数值加一放置在 OUT 的变量中。INC 指令根据操作数不同分 INC _ B 字节加一指令、INC _ W 字加一指令和 INC _ DW 双字加一指令。

问 2：INC 指令样式是怎样的?

答：这些指令的样式分别如图 9-1 ~ 图 9-3 所示。



输入/输出	操作数	数据类型
IN	VB,IB,QB,MB,SB,SMB,LB,AC,常量,*VD,*LD,*AC	字节
OUT	VB,IB,QB,MB,SB,SMB,LB,AC,*VD,*LD,*AC	字节

图 9-1　INC _ B 指令样式



输入/输出	操作数	数据类型
IN	VW,IW,QW,MW,SW,SMW,AC,AIW,LW,T,C,常量,*VD,*LD,*AC	整数
OUT	VW,IW,QW,MW,SW,SMW,LW,AC,T,C,*VD,*LD,*AC	整数

图 9-2　INC _ W 指令样式



输入/输出	操作数	数据类型
IN	VD,ID,QD,MD,SD,SMD,LD,AC,HC,常量,*VD,*LD,*AC	双整数
OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	双整数

图 9-3　INC _ DW 指令样式

问 3：如何应用 INC 指令?

答：INC _ B 指令在输入字节（IN）上加 1，并将结果置入 OUT 指定的变量中，运算不

带符号（16#80 > 16#7F）。

当执行 INC _ B 指令结果为 0 时，零标志位 SM1.0 会变为“1”。

当执行 INC _ B 指令结果从 255 变为 0 时，溢出标志位 SM1.1 会变为“1”、零标志位 SM1.0 会变为“1”。

INC _ W 指令在输入字（IN）上加 1，并将结果置入 OUT 指定的变量中，运算带符号（16#7F00 > 16#8000）。

当执行 INC _ W 指令结果为 0 时，零标志位 SM1.0 会变为“1”。

当执行 INC _ W 指令结果从 32 767 变为 -32 768 时，溢出标志位 SM1.1 会变为“1”。

当执行 INC _ W 指令结果在 -1 ~ -32 768 的范围时，负标志位 SM1.2 会变为“1”。

INC _ DW 指令在输入双字（IN）上加 1，并将结果置入 OUT 指定的变量中，运算带符号（16#7F00 0000 > 16#8000 0000）。

当执行 INC _ DW 指令结果为 0 时，零标志位 SM1.0 会变为“1”。

当执行 INC _ DW 指令结果从 2 147 483 647 变为 -2 147 483 648 时，溢出标志位 SM1.1 会变为“1”。

当执行 INC _ DW 指令结果在 -1 ~ -2 147 483 648 的范围时，负标志位 SM1.2 会变为“1”。

INC 加“1”指令数值变化规律如图 9-4 所示。

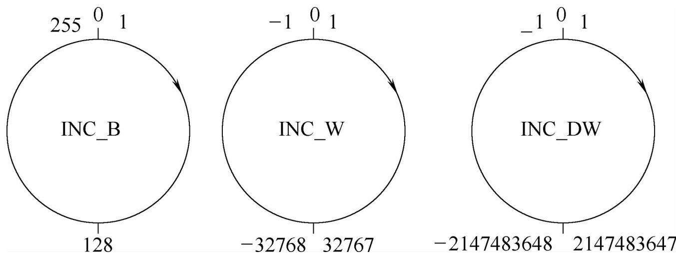


图 9-4　INC 指令规律

如图 9-5 所示程序，当 M0.1 接通时，输入与输出、特殊继电器标志位及 ENO 的状态见表 9-1。

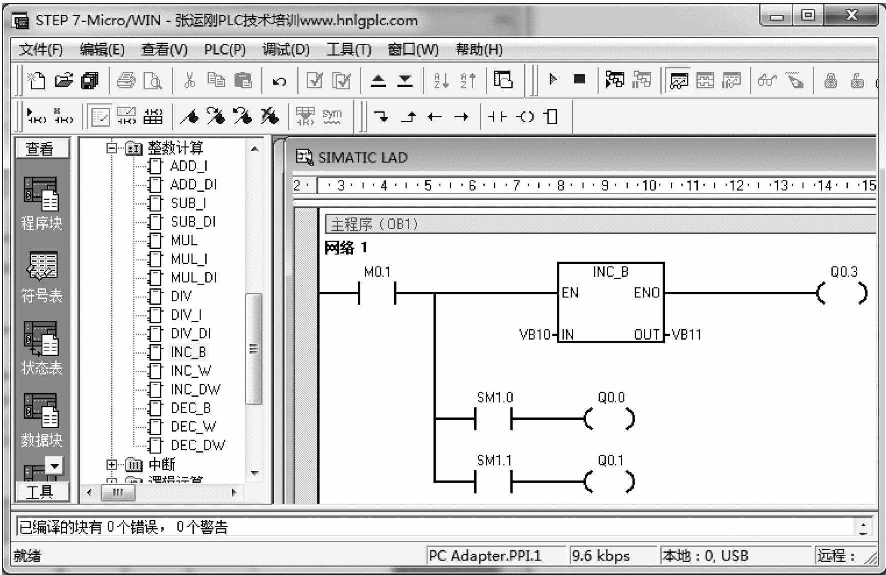


图 9-5　INC _ B 指令例 1

表 9-1　INC _ B 指令状态

VB10 数值	VB11 数值	零标志位 SM1. 0(Q0. 0)	溢出标志位 SM1. 1(Q0. 1)	ENO(Q0. 3)
0	1	0	0	1
1	2	0	0	1
2	3	0	0	1
……	……	……	……	……
255	0	1	1	0
0	1	0	0	1

图 9-6 所示的程序中，源和目标相同，当 M0. 1 接通，VB10 的数值每个扫描周期都在加 1。这种情况考虑使用脉冲执行型指令或确保驱动信号只接通一个扫描周期的时间，如图 9-7 所示。

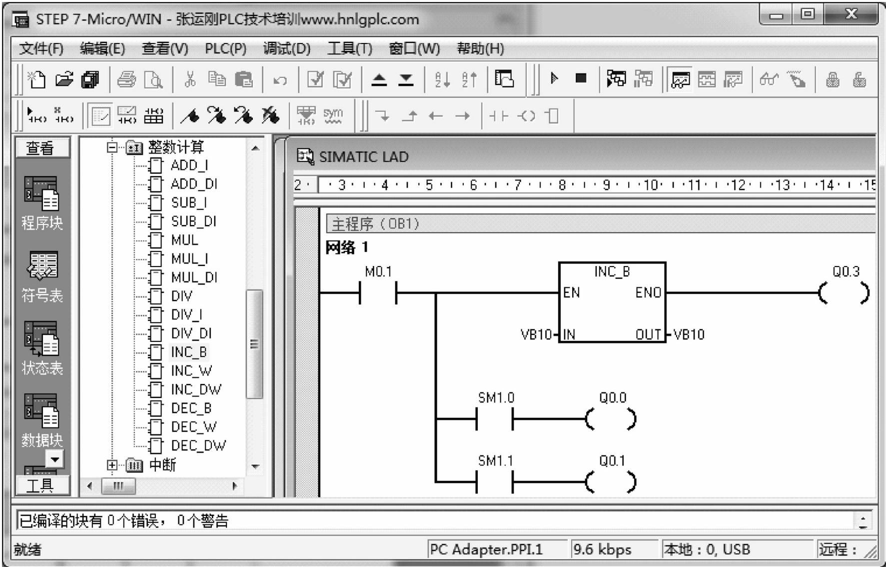


图 9-6　INC _ B 指令例 2

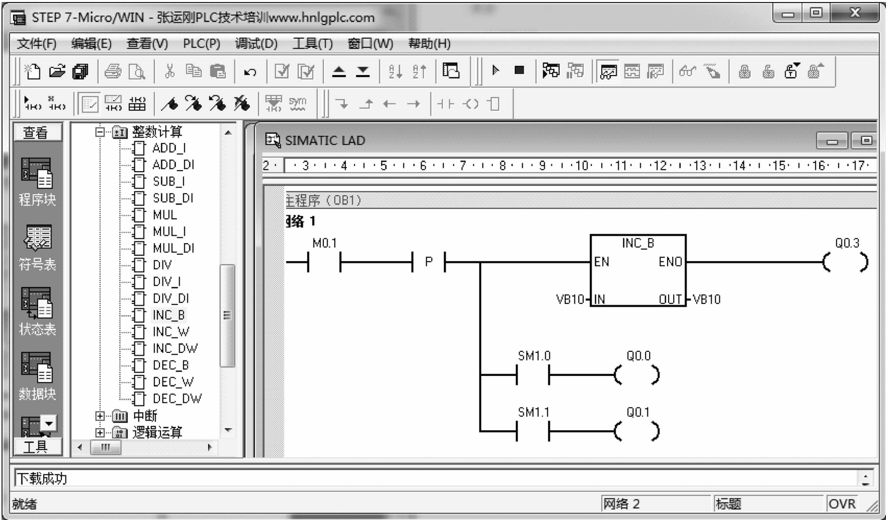


图 9-7　INC _ B 指令例 3

如图 9-8 所示程序，当 M0. 1 接通时，输入与输出、特殊继电器标志位及 ENO 的状态见表 9-2。

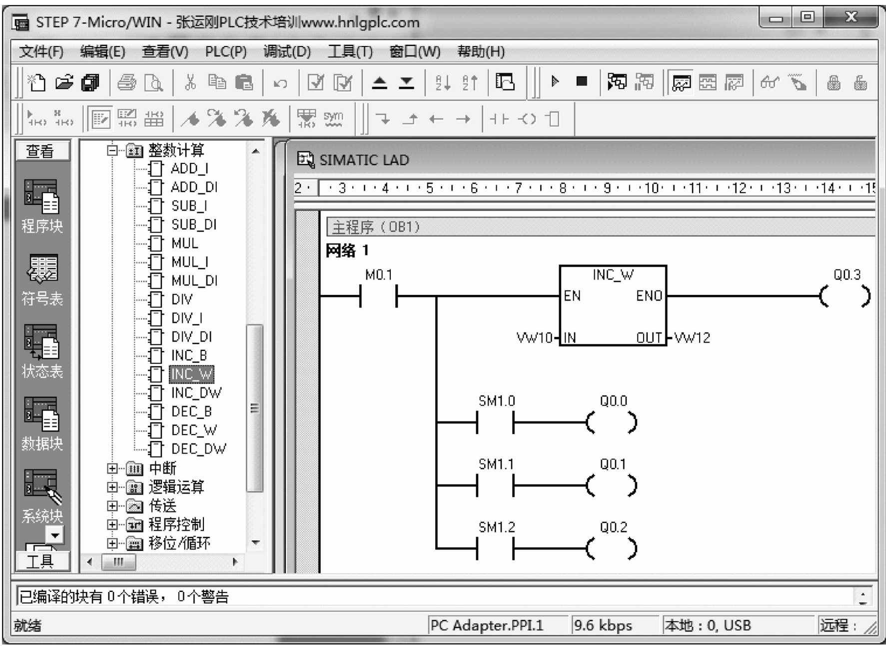


图 9-8　INC _ W 指令例

表 9-2　INC _ W 指令状态

VW10 数值	VW12 数值	零标志位 SM1. 0(Q0. 0)	溢出标志位 SM1. 1(Q0. 1)	负标志位 SM1. 2(Q0. 2)	ENO(Q0. 3)
0	1	0	0	0	1
1	2	0	0	0	1
……	……	……	……	……	……
32 766	32 767	0	0	0	1
32 767	-32 768	0	1	1	0
-32 768	-32 767	0	0	1	1
……	……	……	……	……	……
-1	0	1	0	0	1
0	1	0	0	0	1

如图 9-9 所示程序，当 M0. 1 接通时，输入与输出、特殊继电器标志位及 ENO 的状态见表 9-3。

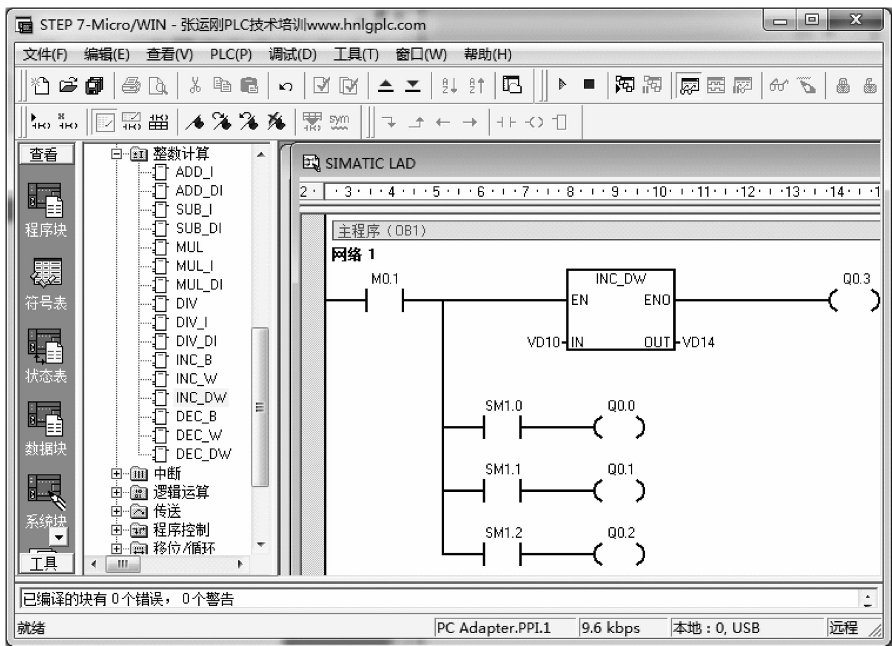


图 9-9　INC _ DW 指令例

表 9-3　INC _ DW 指令状态

VD10 数值	VD14 数值	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	ENO(Q0.3)
0	1	0	0	0	1
1	2	0	0	0	1
……	……	……	……	……	……
2 147 483 646	2 147 483 647	0	0	0	1
2 147 483 647	-2 147 483 648	0	1	1	0
-2 147 483 648	-2 147 483 647	0	0	1	1
……	……	……	……	……	……
-1	0	1	0	0	1
0	1	0	0	0	1

在图 9-10 所示的指令中，M0.0 在未接通及接通第一次、第二次、第三次、第四次、第五次、第六次时：VD20 的数值分别指向 T0、T1、T2、T3、T4、T5、T0 的地址，VW30 的数值分别代表 T0、T1、T2、T3、T4、T5、T0 的经过值。

问 4：在应用 INC 指令时需要注意些什么?

答：在使用 INC 指令时需注意以下两点：

一是间接寻址时注意不要超出寻址范围。

二是注意脉冲执行和连续执行是有区别的。

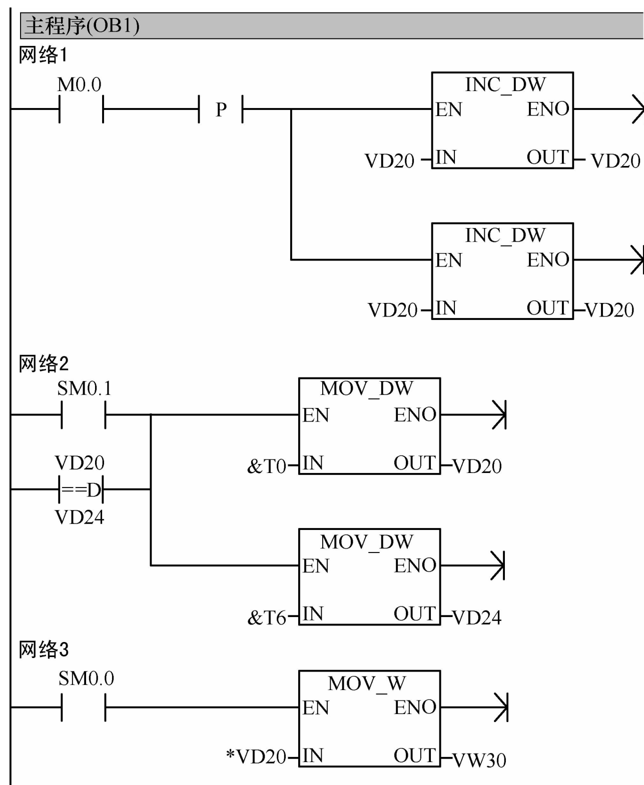


图 9-10　INC _ DW 指令应用

9.2　DEC _ B/W/DW

> 问：DEC 指令基本功能是什么？
> DEC 指令样式是怎样的？
> 如何应用 DEC 指令？
> 在应用 DEC 指令时需要注意些什么？

问 1：DEC 指令基本功能是什么？

答：执行 DEC 一次，IN 中的变量数值减少一放置在 OUT 的变量中。DEC 指令根据操作数不同分 DEC _ B 字节减一指令、DEC _ W 字减一指令和 DEC _ DW 双字减一指令。

问 2：DEC 指令样式是怎样的？

答：DEC 指令的样式分别如图 9-11 ~ 图 9-13 所示。



图 9-11　DEC _ B 指令样式



图 9-12　DEC _ W 指令样式

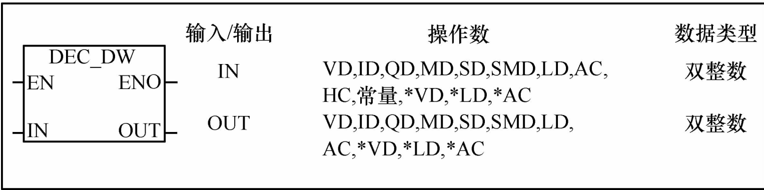


图 9-13　DEC _ DW 指令样式

问 3：如何应用 DEC 指令?

答：DEC 指令的规律和应用如下：

当执行 DEC _ B 指令结果为 0 时，零标志位 SM1. 0 会变为“1”。

当执行 DEC _ B 指令结果从 0 变为 255 时，溢出标志位 SM1. 1 会变为“1”、零标志位 SM1. 0 会变为“1”。

DEC _ W 指令在输入字（IN）上减 1，并将结果置入 OUT 指定的变量中，运算带符号（16#7F00 > 16#8000）。

当执行 DEC _ W 指令结果为 0 时，零标志位 SM1. 0 会变为“1”。

当执行 DEC _ W 指令结果从 −32 768 变为 32 767 时，溢出标志位 SM1. 1 会变为“1”。

当执行 DEC _ W 指令结果在 −1 ~ −32 768 的范围时，负标志位 SM1. 2 会变为“1”。

DEC _ DW 指令在输入双字（IN）上减 1，并将结果置入 OUT 指定的变量中，运算带符号（16#7F00 0000 > 16#8000 0000）。

当执行 DEC _ DW 指令结果为 0 时，零标志位 SM1. 0 会变为“1”。

当执行 DEC _ DW 指令结果从 −2 147 483 648 变为 2 147 483 647 时，溢出标志位 SM1. 1 会变为“1”。

当执行 DEC _ DW 指令结果在 −1 ~ −2 147 483 648 的范围时，负标志位 SM1. 2 会变为“1”。

DEC 减“1”指令数值变化规律如图 9-14 所示。

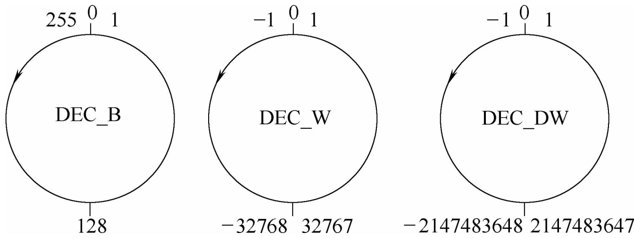


图 9-14　DEC 减“1”指令规律

如图 9-15 所示程序，当 M0.1 接通时，输入与输出、特殊继电器标志位及 ENO 的状态见表 9-4。

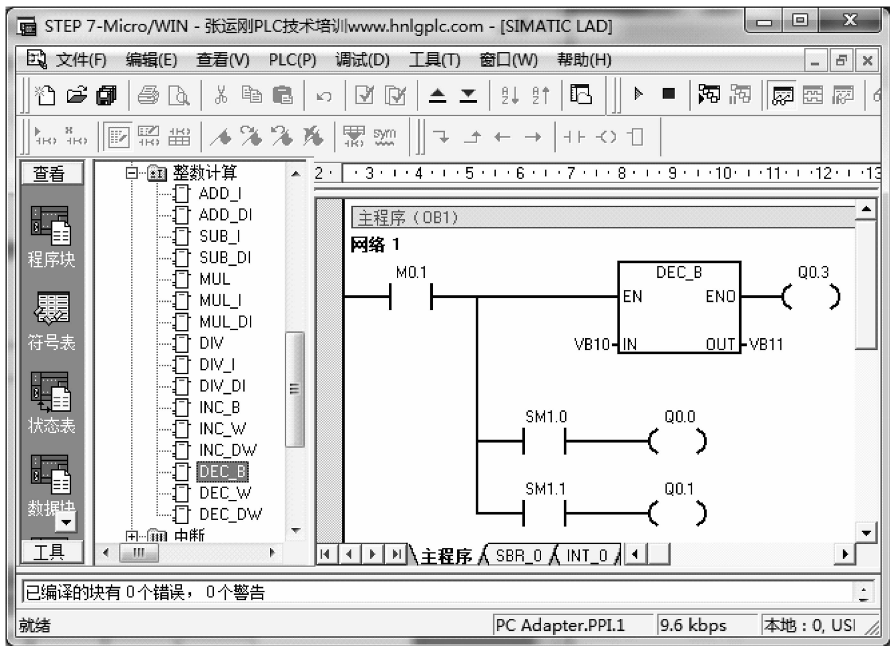


图 9-15　DEC _ B 指令例 1

表 9-4　DEC _ B 指令状态

VB10 数值	VB11 数值	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	ENO(Q0.3)
0	255	0	1	0
255	254	0	0	1
254	253	0	0	1
……	……	……	……	……
1	0	1	0	1
0	255	0	1	0

图 9-16 所示的程序中，源和目标相同时，当 M0.1 接通，VB10 的数值每个扫描周期都在减 1。这种情况考虑使用脉冲执行型指令或确保驱动信号只接通一个扫描周期的时间，如图 9-17 所示。

如图 9-18 所示程序，当 M0.1 接通时，输入与输出、特殊继电器标志位及 ENO 的状态见表 9-5。

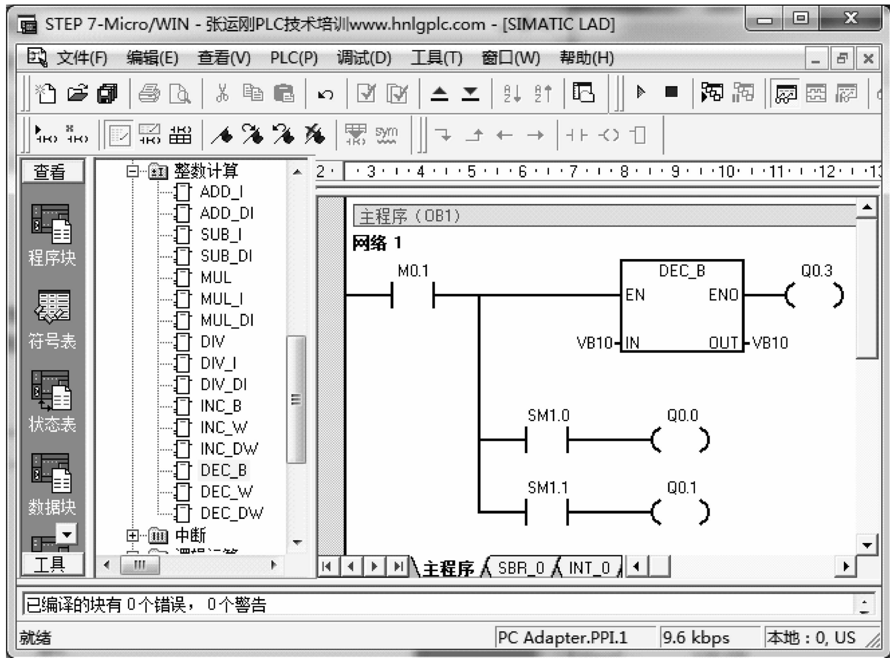


图 9-16　DEC _ B 指令例 2

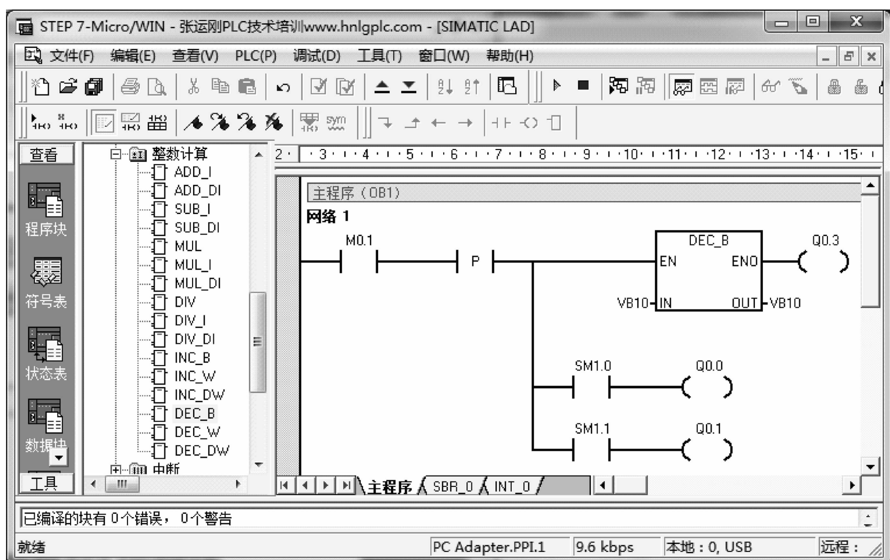


图 9-17　DEC _ B 指令例 3

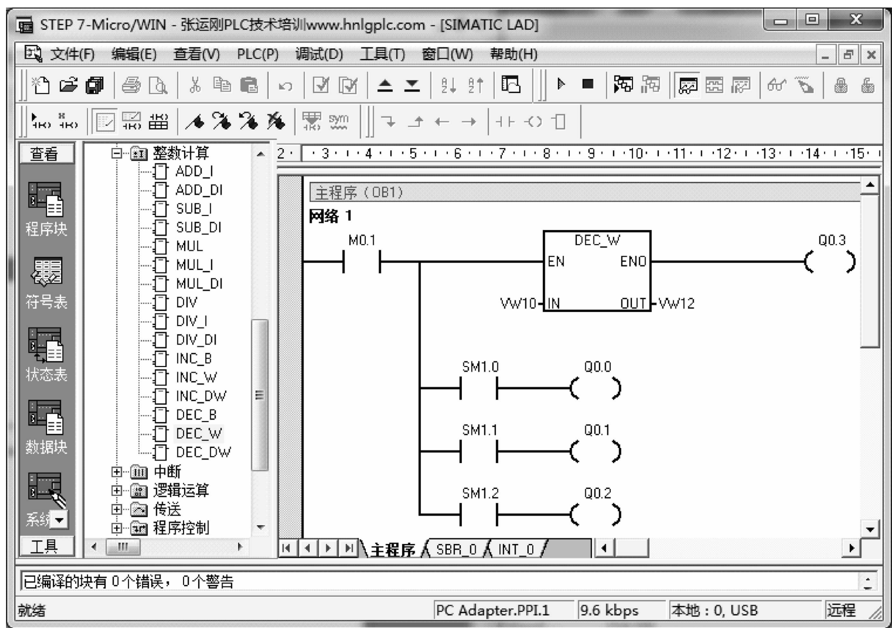


图 9-18　DEC _ W 指令例

表 9-5　DEC _ W 指令状态

VW10 数值	VW12 数值	零标志位 SM1. 0(Q0. 0)	溢出标志位 SM1. 1(Q0. 1)	负标志位 SM1. 2(Q0. 2)	ENO(Q0. 3)
0	-1	0	0	1	1
-1	-2	0	0	1	1
……	……	……	……	……	……
-32 767	-32 768	0	0	1	1
-32 768	32 767	0	1	0	0
32 767	32 766	0	0	0	1
……	……	……	……	……	……
1	0	1	0	0	1
0	-1	0	0	1	1

如图 9-19 所示程序，当 M0. 1 接通时，输入与输出、特殊继电器标志位及 ENO 的状态见表 9-6。

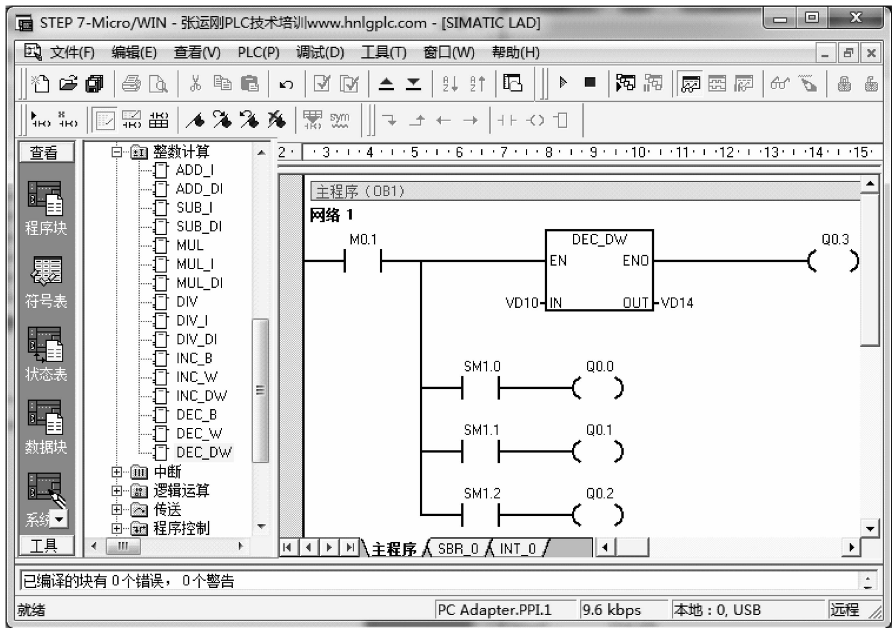


图 9-19　DEC _ DW 指令例

表 9-6　DEC _ DW 指令状态

VD10 数值	VD14 数值	零标志位 SM1.0(Q0.0)	溢出标志位 SM1.1(Q0.1)	负标志位 SM1.2(Q0.2)	ENO (Q0.3)
0	-1	0	0	1	1
-1	-2	0	0	1	1
……	……	……	……	……	……
-2 147 483 647	-2 147 483 648	0	0	1	1
-2 147 483 648	2 147 483 647	0	1	0	0
2 147 483 647	2 147 483 646	0	0	0	1
2 147 483 646	2 147 483 645	0	0	0	1
……	……	……	……	……	……
1	0	1	0	0	1
0	-1	0	0	1	1

在图 9-20 所示的程序中，M0.0 在未接通、接通第一次、第二次、第三次、第四次、第五次、第六次时：VD20 的数值分别指向 T16、T15、T14、T13、T12、T11、T16 的地址，VW30 的数值分别代表 T16、T15、T14、T13、T12、T11、T16 的经过值。

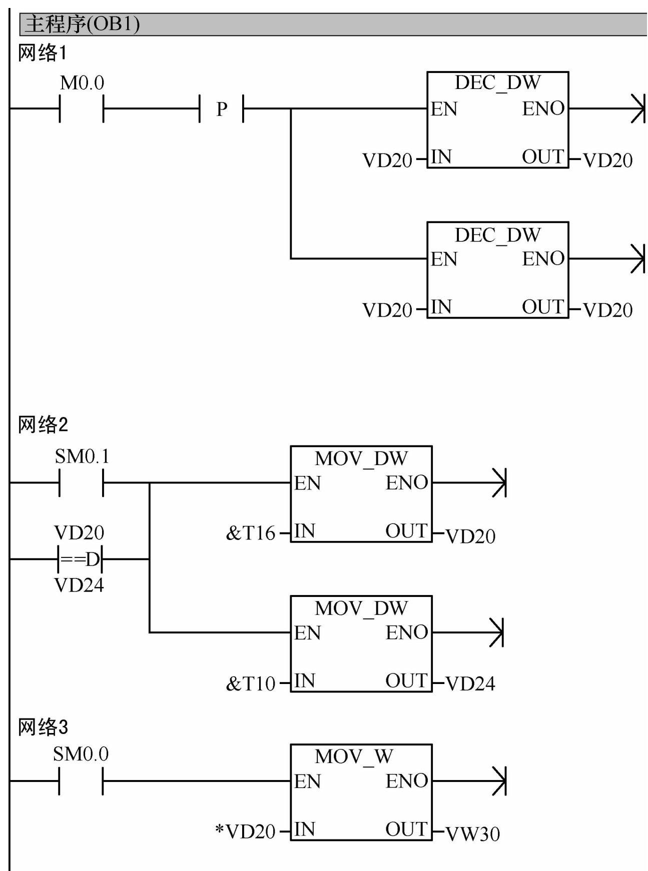


图 9-20　DEC _ DW 指令应用

问 4：在应用 DEC 指令时需要注意些什么？

答：在使用 DEC 指令时需注意以下两点：

一是间接寻址时注意不要超出寻址范围。

二是注意脉冲执行和连续执行是有区别的。

9.3　几种加减法有何不同

问：编程时常用的加/减法逻辑有几种？
这些加/减法各有什么特点？

问 1：编程时常用的加/减法逻辑有几种？

答：在编程时常用的加法有：CTU 加法计数器、ADD 加法运算和 INC 加一指令；常用的减法有：CTD 减法计数器、SUB 减法运算和 DEC 减一指令。

问 2：这些加/减法各有什么特点？

答：这些加减法有各自的特点，描述如下：

CTU 加法计数器的特点是，操作对象是计数器 C，其是在驱动信号上升沿时加计数；CTD 减法计数器的特点是，操作对象也是计数器 C，其是在驱动信号上升沿时减计数。加/减计数器程序如图 9-21 所示，每当 I0.2 接通时 C10 的经过值加一，每当 I0.3 接通时 C11 的经过值减一。

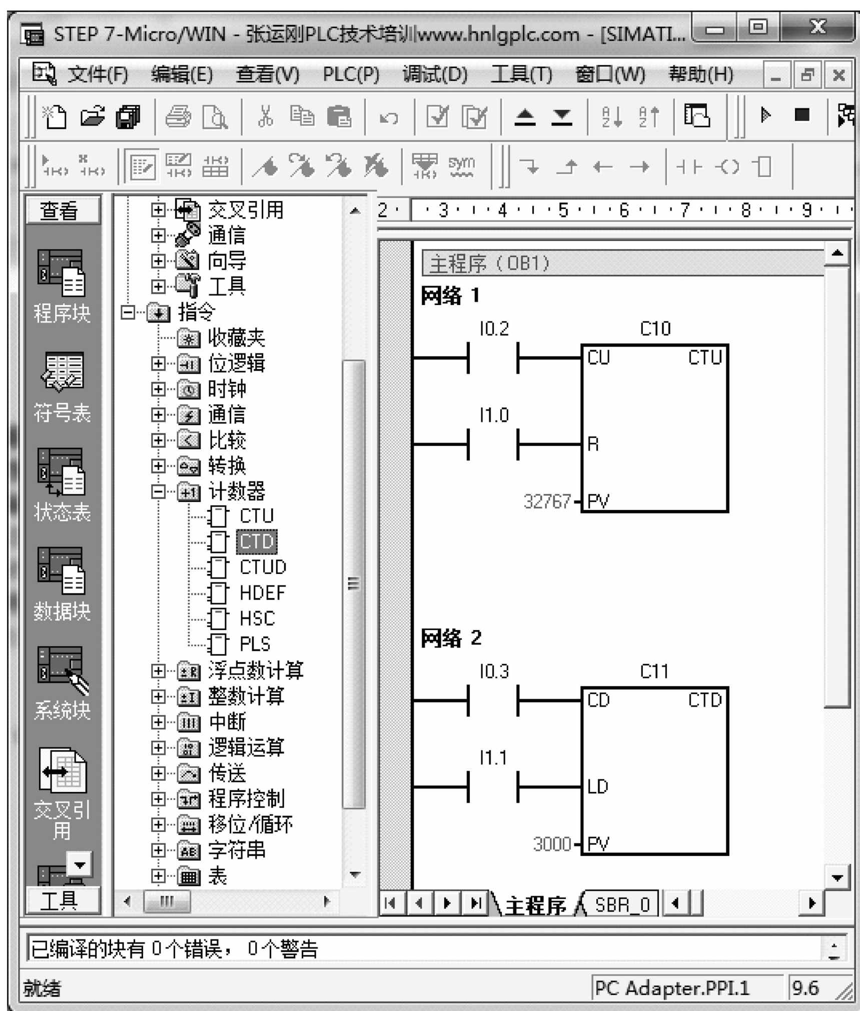


图 9-21　CTU/CTD 计数器程序

ADD 加法运算的特点是，其操作数可以是 I、DI 或者 R，当驱动信号为“1”的每个扫描周期都实现“IN1 + IN2 = OUT”这个运算；SUB 减法运算的特点是，其操作数可以是 I、DI 或者 R，当驱动信号为“1”的每个扫描周期都实现“IN1 – IN2 = OUT”这个运算。

如图 9-22 所示，每当 I0.2 接通时 VW100 的数值加一，每当 I0.3 接通时 VW102 的数值减一。

如图 9-23 所示，每当 I0.2 接通时 VW100 的数值加 3，每当 I0.3 接通时 VW102 的数值减 10。

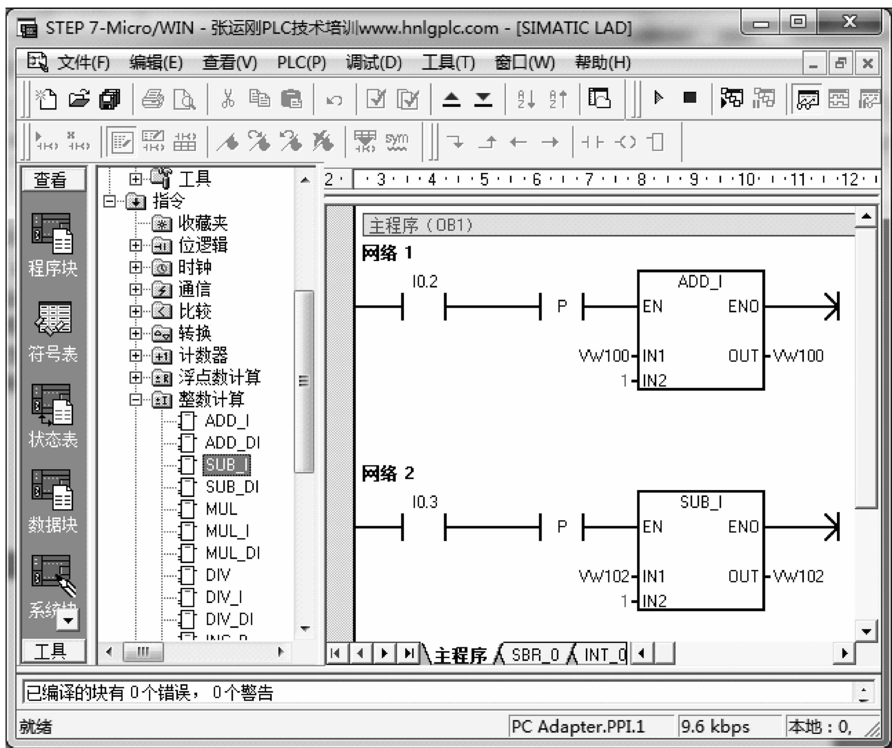


图 9-22　ADD/SUB 实现加一/减一

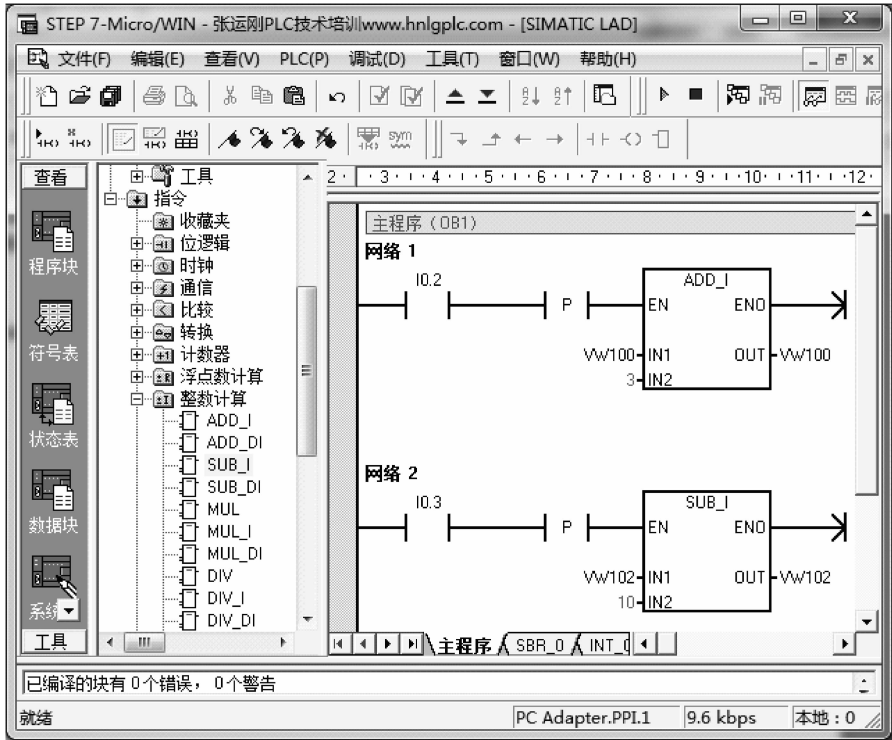


图 9-23　ADD/SUB 实现加/减任意数值

INC 加一指令的特点是，其操作数可以是 B、I 或者 DI，当驱动信号为“1”的每个扫描周期都实现“IN + 1 = OUT”这个运算；DEC 减一指令的特点是，其操作数可以是 B、I、或者 DI，当驱动信号为“1”的每个扫描周期都实现“IN − 1 = OUT”这个运算。

加一/减一程序如图 9-24 所示，每当 I0.2 接通时 VD100 数值加一，每当 I0.3 接通 VW200 数值减一。

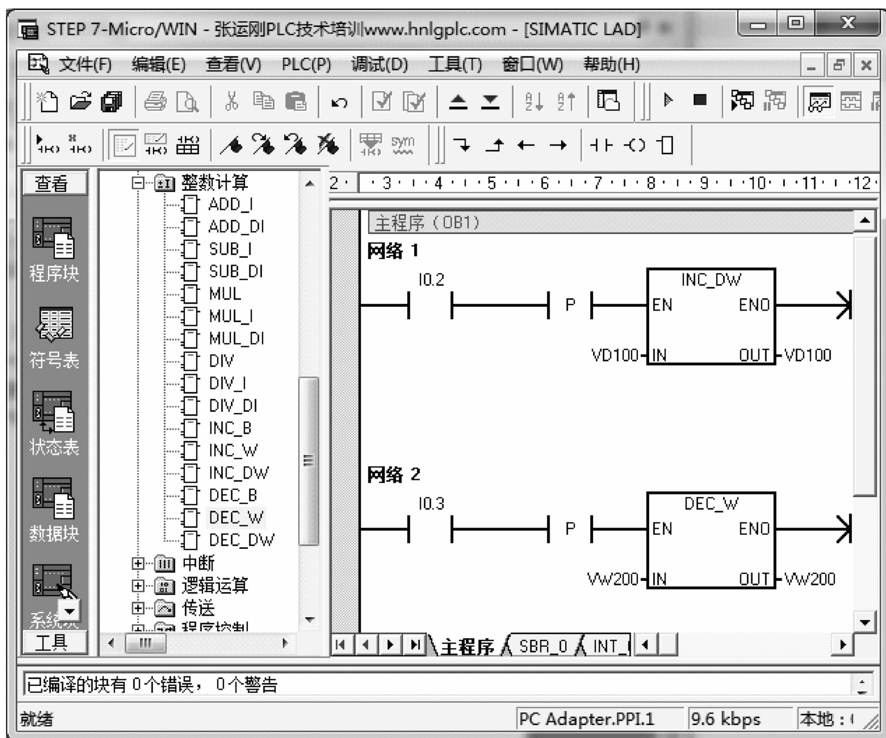


图 9-24　INC/DEC 实现加一/减一

第 10 章　循环移位表逻辑指令

10.1　SHL/SHR _ B/W/DW

> 问：移位指令基本功能是什么？
> 　　移位指令样式是怎样的？
> 　　如何应用移位指令？
> 　　在应用移位指令时需要注意些什么？

问 1：移位指令基本功能是什么？

答：移位指令的基本功能就是实现移位目的。按照移位方向分左移位和右移位，按照操作数不同又分字节、字和双字移位。

问 2：移位指令样式是怎样的？

答：这些移位指令见表 10-1，它们的指令样式如图 10-1 ~ 图 10-6 所示。

表 10-1　移位指令分类表

	字节	字	双字
左移位	SHL _ B	SHL _ W	SHL _ DW
右移位	SHR _ B	SHR _ W	SHR _ DW



图 10-1　SHL _ B 指令样式



图 10-2　SHL _ W 指令样式



图 10-3　SHL _ DW 指令样式

SHR_B	输入/输出	操作数	数据类型
EN ENO	IN	VB,IB,QB,MB,SMB,SB,LB, AC,常量,*VD,*LD,*AC	字节
IN OUT	N	VB,IB,QB,MB,SMB,SB,LB, AC,常量,*VD,*LD,*AC	字节
N	OUT	VB,IB,QB,MB,SMB,SB,LB, AC,*VD,*LD,*AC	字节

图 10-4　SHR _ B 指令样式

SHR_W	输入/输出	操作数	数据类型
EN ENO	IN	VW,T,C,IW,QW,MW,SW,SMW, LW,AC,AIW,常量,*VD*LD,*AC	字
IN OUT	N	VB,IB,QB,MB,SB,SMB,LB,AC, 常量,*VD,*LD,*AC	字节
N	OUT	VW,T,C,IW,QW,MW,SW,SMW, LW,AC,*VD,*LD,*AC	字

图 10-5　SHR _ W 指令样式

SHR_DW	输入/输出	操作数	数据类型
EN ENO	IN	VD,ID,QD,MD,SD,SMD,LD, AC,HC,常量,*VD,*LD,*AC	双字
IN OUT	N	VB,IB,QB,MB,SB,SMB,LB, AC,常量,*VD,*LD,*AC	字节
N	OUT	VD,ID,QD,MD,SD,SMD,LD, AC,*VD,*LD,*AC	双字

图 10-6　SHR _ DW 指令样式

问 3：如何应用移位指令?

答：移位指令应用描述如下：

SHL _ B 字节左移和 SHR _ B 字节右移指令将输入字节（IN）数值的二进制数按照指定移位数（N）向左或向右移动，并将结果载入输出字节（OUT）。移位指令对每个移出位自动补 0。

如果移位数目（N）大于或等于 8，则数值最多被移动 8 位。

如果移位数目大于 0，移出的最后一位状态将复制在 SM1. 1 上。如果移位操作结果为 0，零标志位 SM1. 0 会为“1”。

SHL _ W 字左移和 SHR _ W 字右移指令将输入字（IN）数值的二进制位向左或向右移动 N 位，并将结果载入输出字（OUT）。移位指令对每个移出位自动补 0。

如果移位数目（N）大于或等于 16，则数值最多被移动 16 位。如果移位数目大于 0，移出的最后一位状态复制在 SM1. 1 中。如果移位操作结果为 0，零标志位 SM1. 0 会为“1”。

SHL _ DW 双字左移和 SHR _ DW 双字右移指令将输入双字（IN）数值的二进制位向左或向右移动 N 位，并将结果载入输出双字（OUT）。移位指令对每个移出位自动补 0。

如果移位数目（N）大于或等于 32，则数值最多被移动 32 位。如果移位数目大于 0，移出的最后一位状态复制在 SM1. 1 中。如果移位操作结果为 0，零标志位 SM1. 0 会为“1”。

下面以 SHL _ W 指令为例，说明移位指令的应用。

在图 10-7 所示的程序中：I0. 2 未接通、第一次接通、第二次接通、第三次接通、第四次接通时，字移位指令移动过程如图 10-8 所示，字左移状态见表 10-2。移出位使用“0000”来填补。

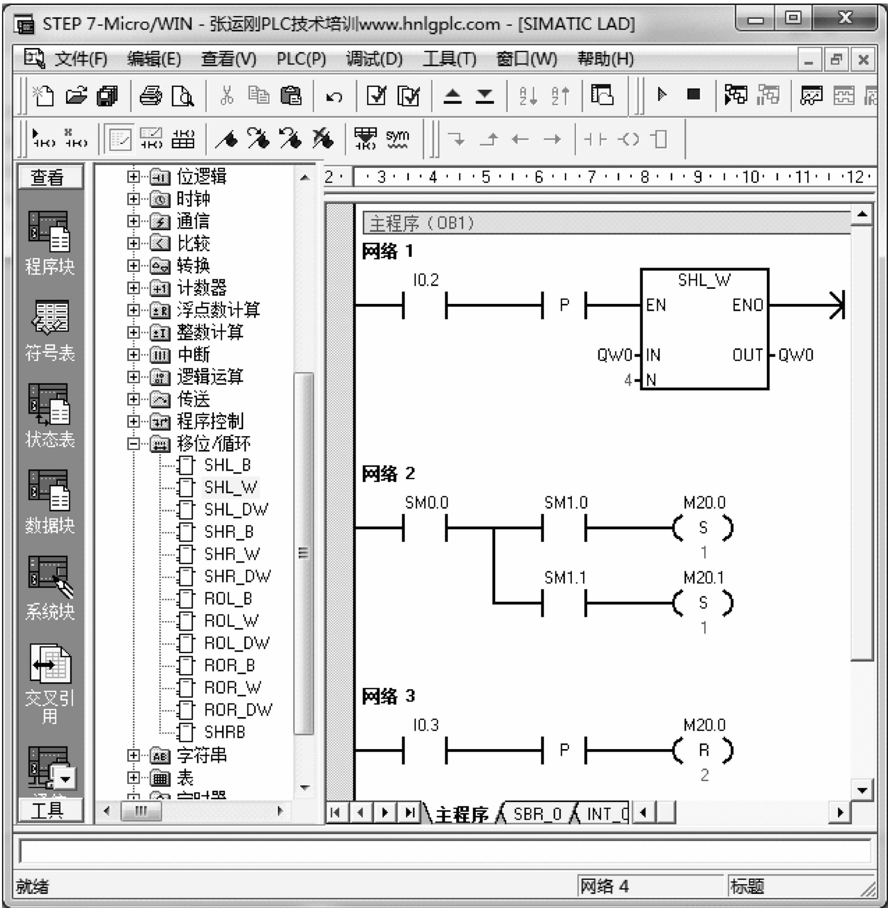


图 10-7　字左移位指令

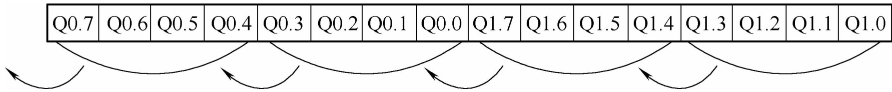


图 10-8　字左移位指令移动过程示意图

表 10-2　字左移状态

QW 0	Q 0. 7	Q 0. 6	Q 0. 5	Q 0. 4	Q 0. 3	Q 0. 2	Q 0. 1	Q 0. 0	Q 1. 7	Q 1. 6	Q 1. 5	Q 1. 4	Q 1. 3	Q 1. 2	Q 1. 1	Q 1. 0	SM 1. 1	SM 1. 0
#0	1	1	0	1	0	1	0	1	1	1	0	0	0	0	1	1	0	0
#1	0	1	0	1	1	1	0	0	0	0	1	1	0	0	0	0	1	
#2	1	1	0	0	0	0	1	1	0	0	0	0	0	0	0	0	1	
#3	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
#4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
#0、#1、#2、#3、#4 分别表示：I0. 2 未接通、第一次接通、第二次接通、第三次接通、第四次接通																		

在图 10-9 所示的程序中：I0.2 未接通、第一次接通、第二次接通、第三次接通、第四次接通，字移位指令应用例移动过程如图 10-10 所示，字左移状态见表 10-3。移出位使用“I1.3、I1.2、I1.1、I1.0”来填补。

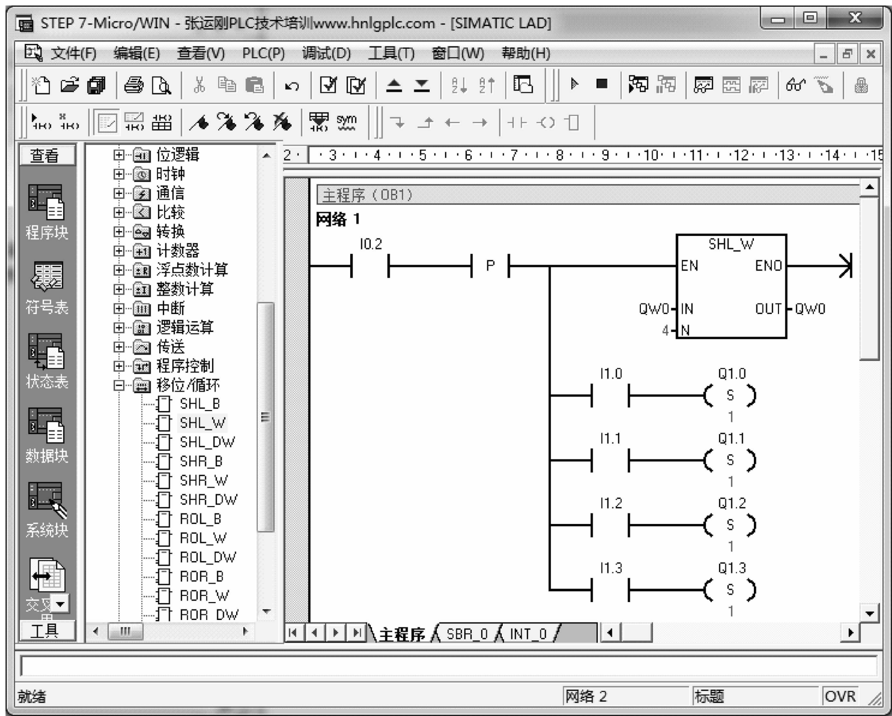


图 10-9　字左移位指令应用例

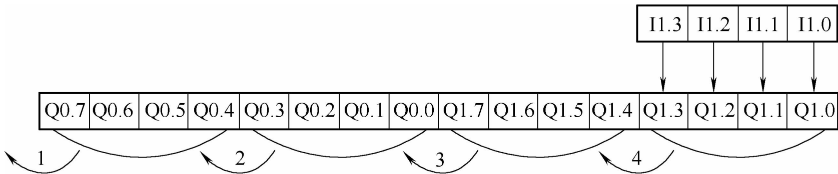


图 10-10　字左移位指令应用例移动过程示意图

表 10-3　字左移状态

QW 0	Q 0.7	Q 0.6	Q 0.5	Q 0.4	Q 0.3	Q 0.2	Q 0.1	Q 0.0	Q 1.7	Q 1.6	Q 1.5	Q 1.4	Q 1.3	Q 1.2	Q 1.1	Q 1.0	I 1.3	I 1.2	I 1.1	I 1.0
#0	1	1	0	1	0	1	0	1	1	1	0	0	0	0	1	1	0	0	1	1
#1	0	1	0	1	1	1	0	0	0	0	1	1	0	1	1	1	0	1	1	1
#2	1	1	0	0	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	1
#3	0	0	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	1	1	0
#4	0	1	1	1	1	1	1	1	1	1	1	0	1	1	0	0	1	1	0	0
#5	1	1	1	1	1	1	1	0	1	1	0	0	1	0	0	0	1	0	0	0
…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…	…
#0、#1、#2、#3、#4、#5 分别表示：I0.2 未接通、第一次接通、第二次接通、第三次接通、第四次接通、第五次接通																				

问 4：在应用移位指令时需要注意些什么？

答：应用移位指令需要注意如下两项：

一是间接寻址时注意不要超出寻址范围。

二是脉冲执行与连续执行是有区别的。

10.2　ROL/ROR _ B/W/DW

问：循环指令基本功能是什么？
　　循环指令样式是怎样的？
　　如何应用循环指令？
　　在应用循环指令时需要注意些什么？

问 1：循环指令基本功能是什么？

答：循环指令的基本功能就是实现循环目的。按照循环方向分左循环和右循环，按照操作数不同又分字节、字和双字循环。

问 2：循环指令样式是怎样的？

答：循环指令见表 10-4 所示，他们的指令样式如图 10-11 ~ 图 10-16 所示。

表 10-4　循环指令分类表

	字节	字	双字
左循环	ROL _ B	ROL _ W	ROL _ DW
右循环	ROR _ B	ROR _ W	ROR _ DW



图 10-11　ROL _ B 指令样式



图 10-12　ROL _ W 指令样式

ROL_DW	输入/输出	操作数	数据类型
EN ENO	IN	VD,ID,QD,MD,SD,SMD,LD,AC,HC,常量,*VD,*LD,*AC	双字
IN OUT	N	VB,IB,QB,MB,SB,SMB,LB,AC,常量,*VD,*LD,*AC	字节
N	OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	双字

图 10-13　ROL _ DW 指令样式

ROR_B	输入/输出	操作数	数据类型
EN ENO	IN	VB,IB,QB,MB,SMB,SB,LB,AC,常量,*VD,*LD,*AC	字节
IN OUT	N	VB,IB,QB,MB,SMB,SB,LB,AC,常量,*VD,*LD,*AC	字节
N	OUT	VB,IB,QB,MB,SMB,SB,LB,AC,*VD,*LD,*AC	字节

图 10-14　ROR _ B 指令样式

ROR_W	输入/输出	操作数	数据类型
EN ENO	IN	VW,T,C,IW,QW,MW,SW,SMW,LW,AC,AIW,常量,*VD,*LD,*AC	字
IN OUT	N	VB,IB,QB,MB,SB,SMB,LB,AC,常量,*VD,*LD,*AC	字节
N	OUT	VW,T,C,IW,QW,MW,SW,SMW,LW,AC,*VD,*LD,*AC	字

图 10-15　ROR _ W 指令样式

ROR_DW	输入/输出	操作数	数据类型
EN ENO	IN	VD,ID,QD,MD,SD,SMD,LD,AC,HC,常量,*VD,*LD,*AC	双字
IN OUT	N	VB,IB,QB,MB,SB,SMB,LB,AC,常量,*VD,*LD,*AC	字节
N	OUT	VD,ID,QD,MD,SD,SMD,LD,AC,*VD,*LD,*AC	双字

图 10-16　ROR _ DW 指令样式

问 3：如何应用循环指令?

答：循环指令应用描述如下：

ROL _ B 字节循环左移和 ROR _ B 字节循环右移指令。将输入字节（IN）数值向左或向右循环移动 N 位，并将结果载入输出字节（OUT）。

如果循环移动数目（N）大于或等于 8，执行循环移动之前先对位数（N）进行除以 8 取模操作，从而使实际循环移动位数在 0 ~ 7 之间。

如果循环移动位数为 0，则不执行循环移动操作。如果循环移动位数为 1 ~ 7，执行循环

移动操作，循环移动的最后一位数值被复制至溢出标志位（SM1.1）。

如果被循环移动的数值为 0，零标志位 SM1.0 会为“1”。

ROL _ W 字循环左移和 ROR _ W 字循环右移指令。将输入字（IN）数值向左或向右循环移动 N 位，并将结果载入输出字（OUT）。

如果循环移动位数（N）大于或等于 16，在循环移动执行之前先对循环移动位数（N）进行除以 16 取模操作，从而使实际循环移动位数在 0 ~ 15 之间。

如果循环移动位数为 0，则不执行循环移动操作。如果循环移动位数为 1 ~ 15，执行循环移动操作，循环移动的最后一位数值被复制至溢出标志位（SM1.1）。

如果被循环移动的数值为 0，零标志位 SM1.0 会为“1”。

ROL _ DW 双字循环左移和 ROR _ DW 双字循环右移指令。将输入双字（IN）数值向左或向右循环移动 N 位，并将结果载入输出双字（OUT）。

如果循环移动位数（N）大于或等于 32，在循环移动执行之前先对循环移动位数（N）进行除以 32 取模操作，从而使实际循环移动位数在 0 ~ 31 之间。

如果循环移动位数为 0，则不执行循环移动操作。如果循环移动位数为 1 ~ 31，执行循环移动操作，循环移动的最后一位数值被复制至溢出标志位（SM1.1）。

如果被循环移动的数值为 0，零标志位 SM1.0 会为“1”。

下面以 ROL _ W 指令为例，说明循环指令的应用。

在图 10-17 所示程序中，VW10 和 SM1.1 的状态见表 10-5，每当 I0.2 接通，VW10 里的二进制数值往左移四位，移出的最后一位状态复制在 SM1.1 中。

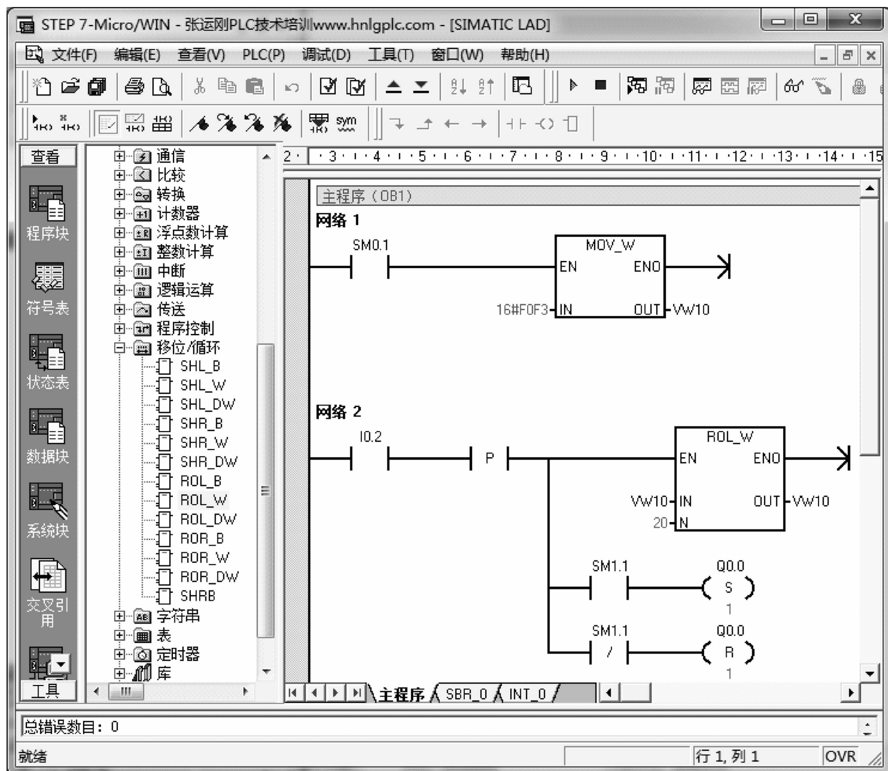


图 10-17　字循环左移

表 10-5　循环左移状态

VW 10	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	SM 1.1
#0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1	0
#1	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	1	1
#2	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0	0	0
#3	0	0	1	1	1	1	1	1	0	0	0	0	1	1	1	1	1
#4	1	1	1	1	0	0	0	0	1	1	1	1	0	0	1	1	1
#5	0	0	0	0	1	1	1	1	0	0	1	1	1	1	1	1	1
……	……																……
#0、#1、#2、#3、#4、#5 分别表示刚运行、I0.2 第一次接通、第二次接通……第五次接通																	

在图 10-18 所示程序中，I0.2 接通次数（循环左移位位数）与 VW10 数值关系见表 10-6 所示。

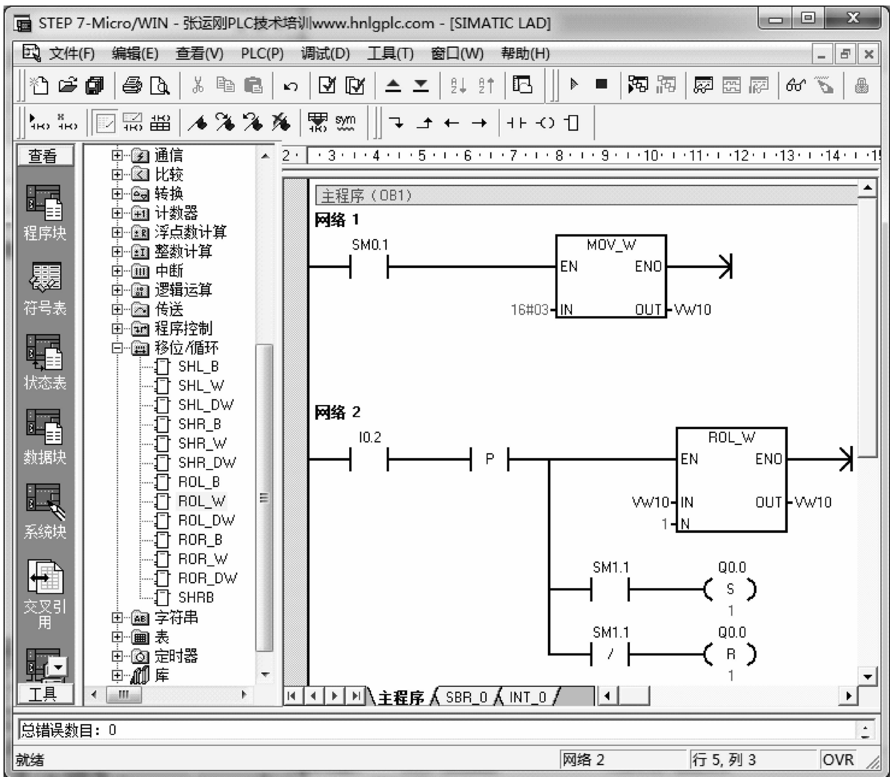


图 10-18　利用 ROL _ W 循环左移实现乘法

表 10-6　循环左移数值关系

I0.0 接通次数	VW10 十进制无符号数	SM1.1	I0.0 接通次数	VW10 十进制无符号数	SM1.1
0	3	0	……	……	……
1	6	0	13	24 576	0
2	12	0	14	49 152	0
3	24	0	15	32 769	1
4	48	0	16	3	1
5	96	0			

从表 10-6 可以看出，在没有溢出时，往左移一位相当于乘 2（2^1），往左移两位相当于乘 4（2^2），往左移三位相当于乘 8（2^3）……往左移 n 位相当于乘以 2^n。

问 4：在应用循环指令时需要注意些什么?

答：应用循环指令需要注意如下两项：

一是间接寻址时注意不要超出寻址范围。

二是脉冲执行与连续执行是有区别的。

10.3　SHRB

问：SHRB 指令基本功能是什么?
SHRB 指令样式是怎样的?
如何应用 SHRB 指令?
在应用 SHRB 指令时需要注意些什么?

问 1：SHRB 指令基本功能是什么?

答：SHRB 位移位的基本功能是实现将 DATA 的位域状态移入位寄存器 S _ BIT 中，同时位寄存器 S _ BIT 里面由 N 指定的位域长度的位也在移动，移动方向由 N 的正负号决定，“N = 正数”左移位，“N = 负数”右移位。SHRB 指令移出的每个位被传送到溢出标志位（SM1.1）中。

问 2：SHRB 指令样式是怎样的?

答：SHRB 位移位指令样式，如图 10-19 所示。



图 10-19　位移位寄存器指令

问 3：如何应用 SHRB 指令？

答：在图 10-20 所示的程序中，I0. 0 未接通、第一次接通、第二次接通……第九次接通，字移位指令应用例移动过程如图 10-21 所示，位移位寄存器指令动作状态见表 10-7。移出位使用“I0. 1”来填补。

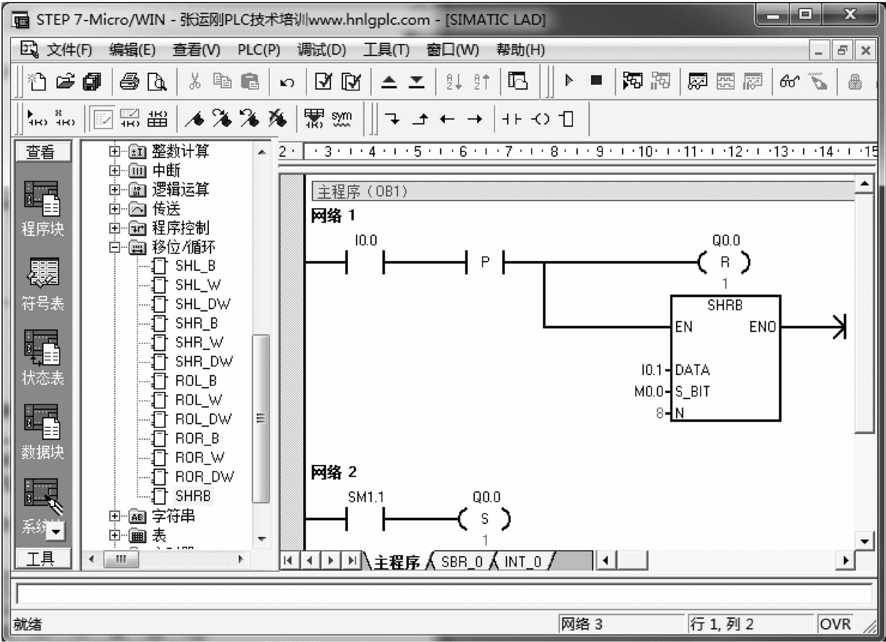


图 10-20　位移位寄存器指令应用例

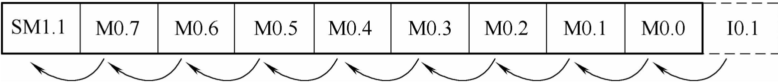


图 10-21　位移位寄存器指令动作示意图

表 10-7　SHRB 指令动作状态

I0. 0	SM1. 1	M0. 7	M0. 6	M0. 5	M0. 4	M0. 3	M0. 2	M0. 1	M0. 0	I0. 1
0*	0	0	0	0	0	0	0	0	0	1
1*	0	0	0	0	0	0	0	0	1	0
2*	0	0	0	0	0	0	0	1	0	1
3*	0	0	0	0	0	0	1	0	1	0
4*	0	0	0	0	0	1	0	1	0	0
5*	0	0	0	0	1	0	1	0	0	1
6*	0	0	0	1	0	1	0	0	1	0
7*	0	0	1	0	1	0	0	1	0	1
8*	0	1	0	1	0	0	1	0	1	0
9*	1	0	1	0	0	1	0	1	0	1
…	…	…	…	…	…	…	…	…	…	…

0＊、1＊、2＊、…、9＊分别表示：I0. 0 未接通、第一次接通、第二次接通……第九次接通

在图 10-22 的应用例程序中，当 I0. 0 ~ I0. 7 依次接通，Q0. 0 ~ Q0. 7 会依次接通，如果顺序搞错，不会动作。

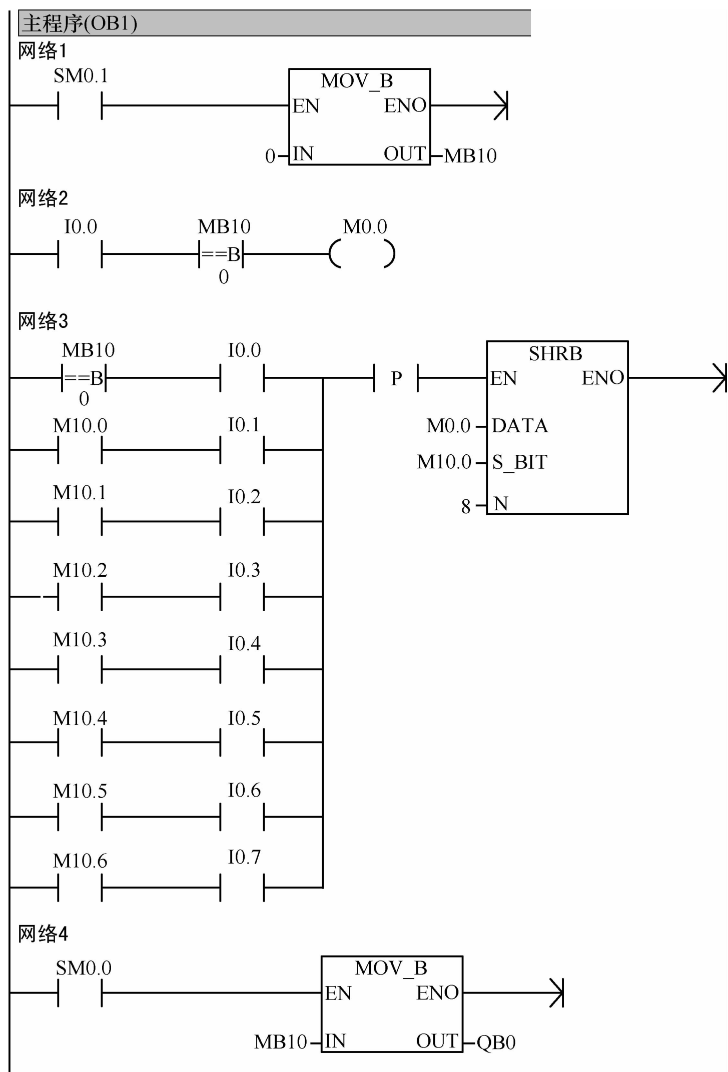


图 10-22　位移位寄存器指令应用例

问 4：在应用 SHRB 指令时需要注意些什么?

答：应用位移位寄存器指令需要注意如下两项：

一是间接寻址时注意不要超出寻址范围。

二是脉冲执行与连续执行是有区别的。

10.4　表指令

问：表指令基本功能是什么?

表指令样式是怎样的?

如何应用表指令?

在应用表指令时需要注意些什么?

问 1：表指令基本功能是什么？

答：表指令包括 AD _ T _ TBL 填表、FIFO 先进先出读表、LIFO 后进先出读表和 TBL _ FIND 查表指令。AD _ T _ TBL 填表指令基本功能是把数据按照前后次序填写入指定表格上，FIFO 先进先出读表指令基本功能是从指定表格里读出最先填写进去的数据，LIFO 后进先出读表指令基本功能是从指定表格里读出最后填写进去的数据，TBL _ FIND 查表指令基本功能是在指定表格里查找满足条件的数据。

问 2：表指令样式是怎样的？

答：表指令样式如图 10-23 ~ 图 10-26 所示。



输入/输出	操作数	数据类型
DATA	VW,IW,QW,MW,SW,SMW,LW,T,C,AIW,AC,常量,*VD,*LD,*AC	整数
TBL	VW,IW,QW,MW,SW,SMW,LW,T,C,*VD,*LD,*AC	字

图 10-23　AD _ T _ TBL 指令样式



输入/输出	操作数	数据类型
TBL	VW,IW,QW,MW,SW,SMW,LW,T,C,*VD,*LD,*AC	字
DATA	VW,IW,QW,MW,SW,SMW,LW,AC,T,C,AQW,*VD,*LD,*AC	整数

图 10-24　FIFO 指令样式



输入/输出	操作数	数据类型
TBL	VW,IW,QW,MW,SW,SMW,LW,T,C,*VD,*LD,*AC	字
DATA	VW,IW,QW,MW,SW,SMW,LW,AC,T,C,AQW,*VD,*LD,*AC	整数

图 10-25　LIFO 指令样式



输入/输出	操作数	数据类型
TBL	VW,IW,QW,MW,SW,SMW,LW,T,C,*VD,*LD,*AC	字
PTN	VW,IW,QW,MW,SW,SMW,AIW,LW,T,C,AC,常量,*VD,*LD,*AC	整数
INDX	VW,IW,QW,MW,SW,SMW,LM,T,C,AC,*VD,*LD,*AC	字
CMD	常量	字节

图 10-26　TBL _ FIND 指令样式

问 3：如何应用表指令？

答：表指令应用描述如下：

在图 10-27 所示的填表指令中，CPU 每次进入 RUN 状态，使用开机脉冲 SM0. 1 把“5”装载入 VW100 中，意思是定义表格的长度为 5。

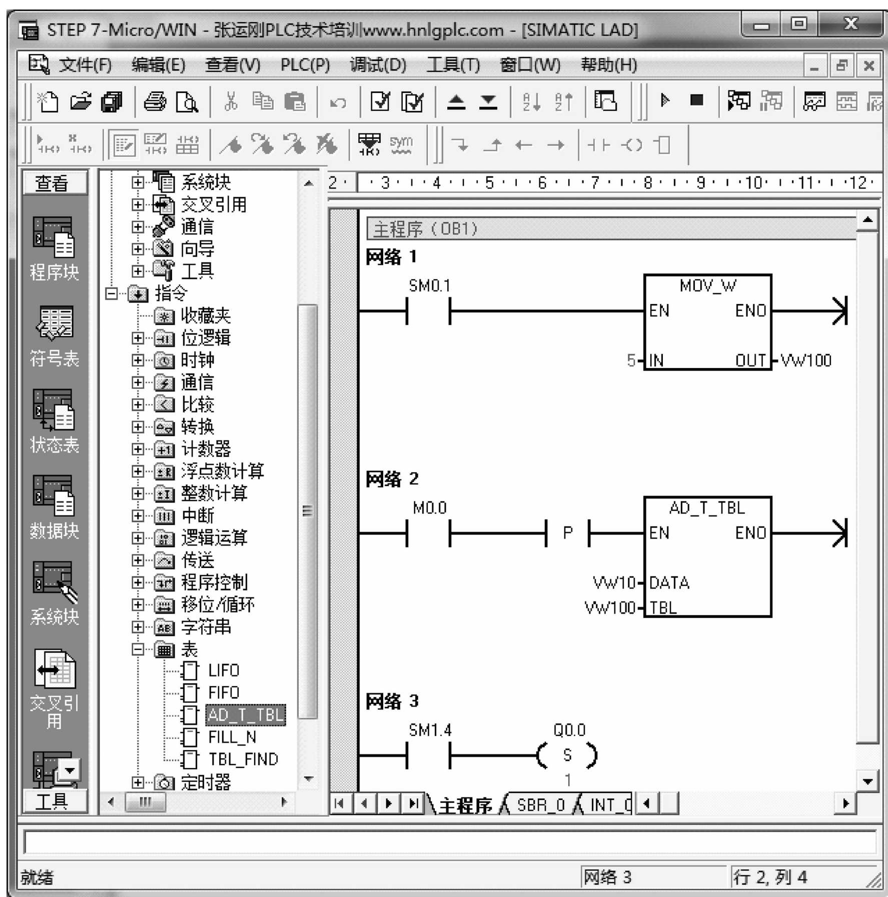


图 10-27　AD _ T _ TBL 填表指令应用

每接通一次 M0.0，都会把 VW10 的当前数值填写入表格中，动作示意图如图 10-28 所示。

当 VW10 = 10、VW102 实际填表数为 0 时，接通一次 M0.0，则 VW104 = 10、VW102 = 1；

当 VW10 = 20、VW102 实际填表数为 1 时，接通一次 M0.0，则 VW106 = 20、VW102 = 2；

当 VW10 = 25、VW102 实际填表数为 2 时，接通一次 M0.0，则 VW108 = 25、VW102 = 3；

当 VW10 = 15、VW102 实际填表数为 3 时，接通一次 M0.0，则 VW110 = 15、VW102 = 4；

当 VW10 = 30、VW102 实际填表数为 4 时，接通一次 M0.0，则 VW112 = 30、VW102 = 5；

当 VW10 = 44、VW102 实际填表数为 5 时，接通一次 M0.0，则表格里的数值没有变化，观察 Q0.0 的状态为 1，填表动作无效。

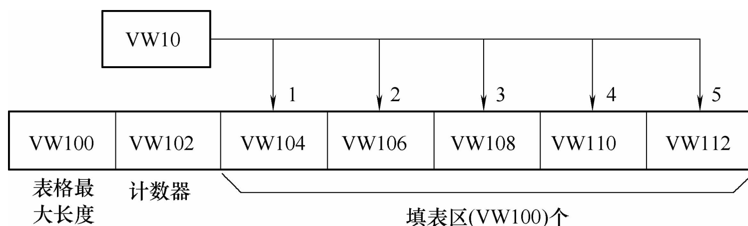


图 10-28　AD _ T _ TBL 填表指令动作示意图

FIFO 指令是读取表格数据，每次执行 FIFO 指令把表格中相对最先写入的数值读出，同时表格实际填表数会减 1。当尝试向一个空表读取数据时，SM1.5 会为 1，表示读取表格动作无效。

在图 10-29 所示程序中，先使用 AD _ T _ TBL 填表指令填写如图 10-30 所示的表格数据，然后使用 FIFO 指令读取表格，M0.1 第一次接通至第六次接通各次的状态依次如图 10-31 ~ 图 10-36 所示。

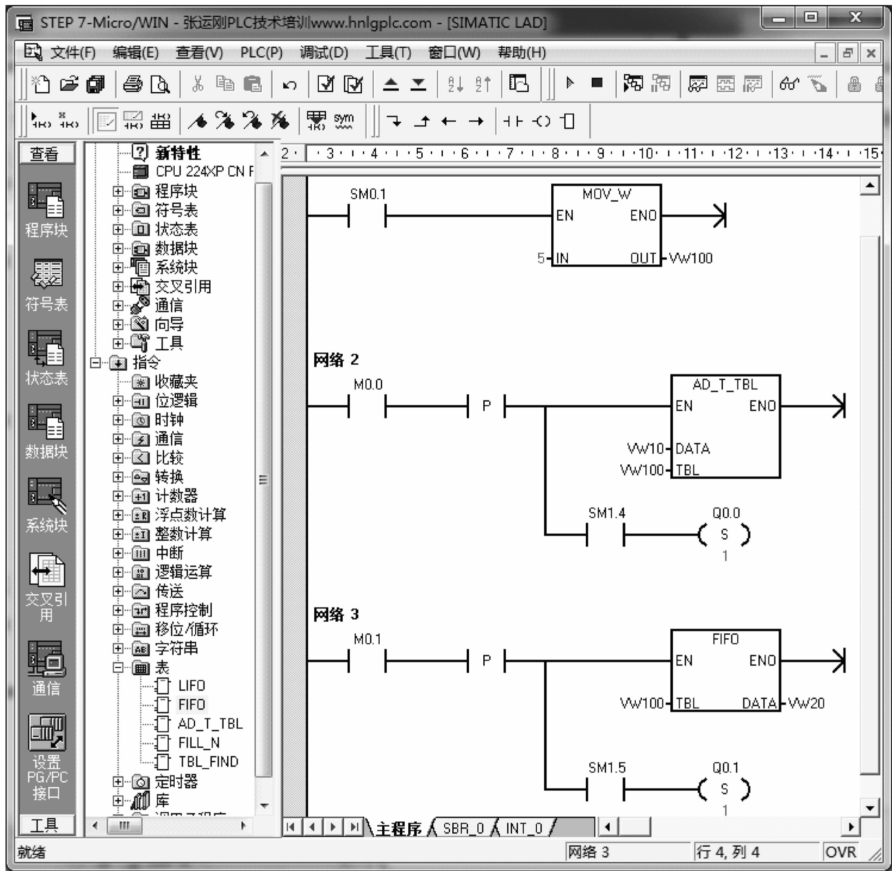


图 10-29　FIFO 指令应用

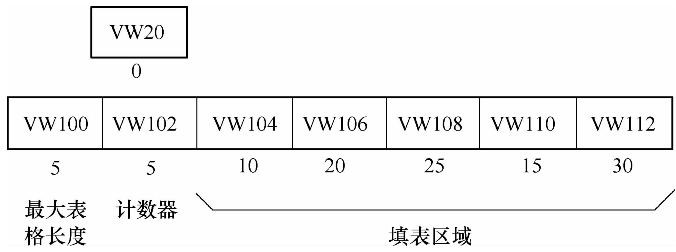


图 10-30　填写的原始表格

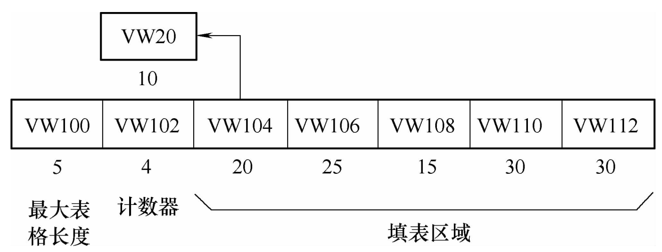


图 10-31　M0.1 接通第 1 次后

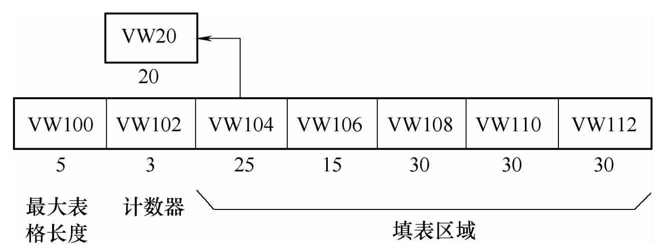


图 10-32　M0.1 接通第 2 次后

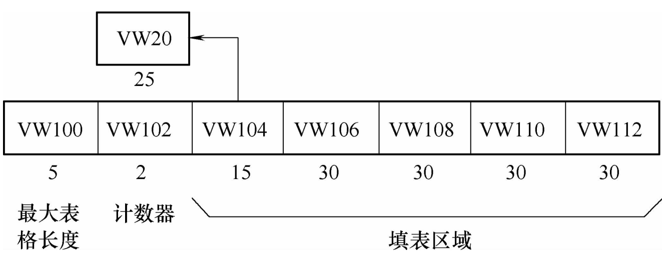


图 10-33　M0.1 接通第 3 次后

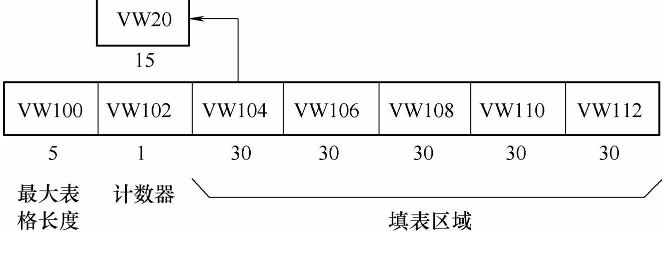


图 10-34　M0.1 接通第 4 次后

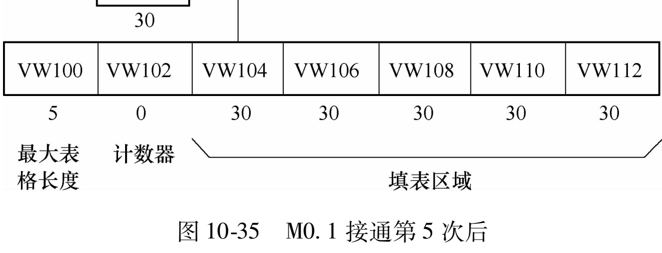


图 10-35　M0.1 接通第 5 次后

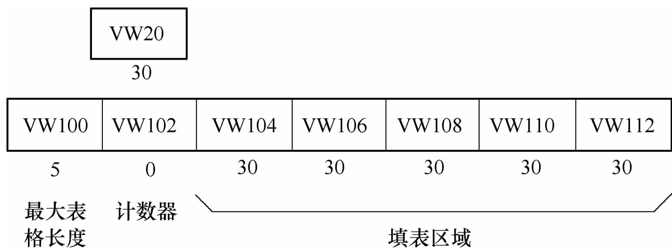


图 10-36　M0. 1 接通第 6 次后

在图 10-37 所示程序中，先使用 AD _ T _ TBL 填表指令填写如图 10-38 所示的表格数据，使用 LIFO 指令读取表格，M0. 1 第一次接通至第五次接通各次的状态依次如图 10-39 ~ 图 10-43 所示。

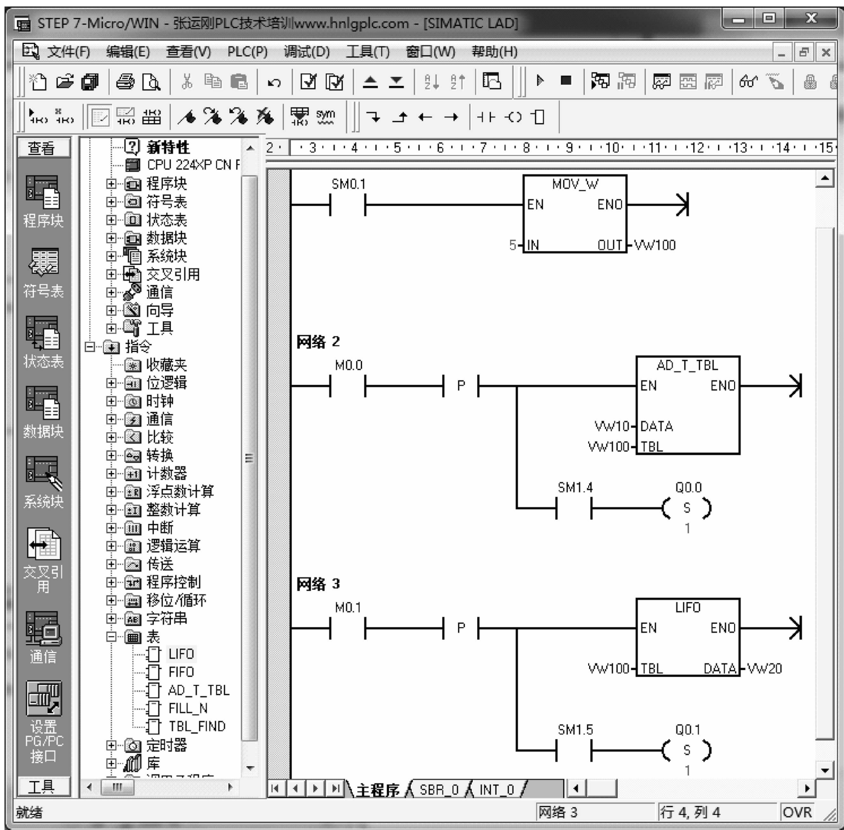


图 10-37　LIFO 指令应用

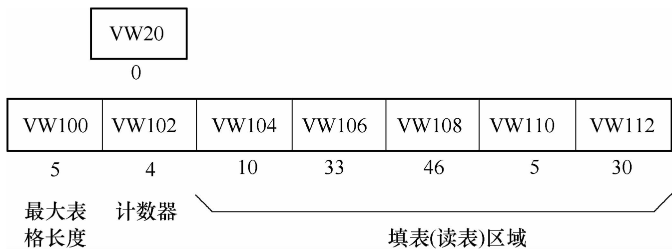


图 10-38　填写的原始表格

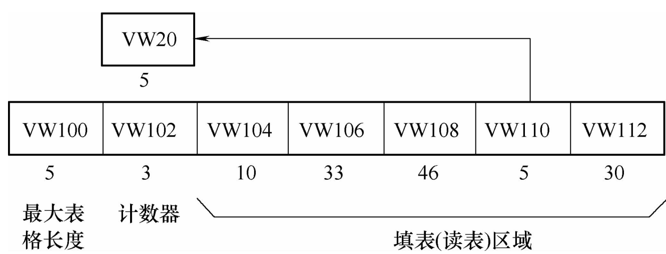


图 10-39　M0.1 接通第 1 次后

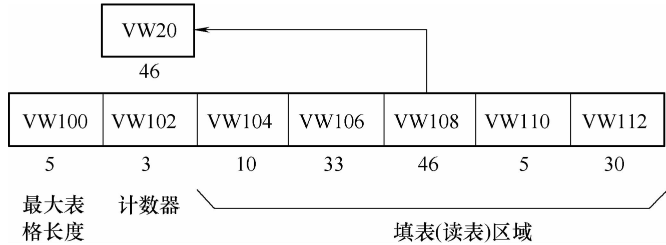


图 10-40　M0.1 接通第 2 次后

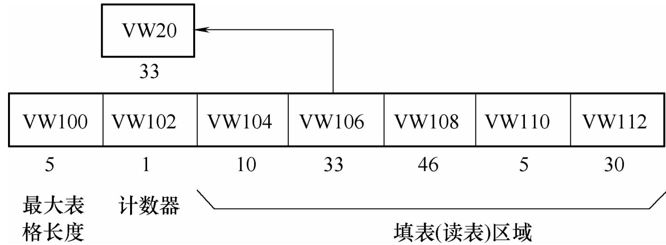


图 10-41　M0.1 接通第 3 次后

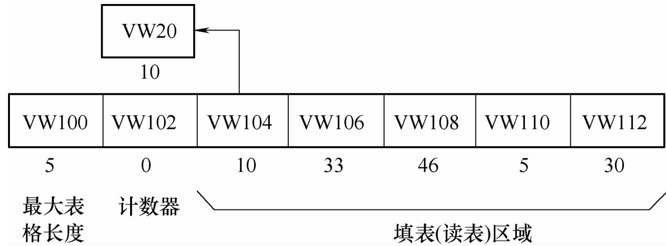


图 10-42　M0.1 接通第 4 次后

TBL _ FIND 指令是从 INDX 指定的使用 AD _ T _ TBL 填表指令填写的表格 TBL 的位置开始，查找与 PTN 指定的数值符合 CMD 匹配的数值的位置，结果存放在 INDX 中。

如果要求从表格的开始位置查找，在开始查找前，先复位 INDX，然后再查找。当查找到一个以后，需要继续往下查找，需要使 INDX 加 1，然后再查找。如果 INDX 的数值等于实际填表数的数值，说明表格查完，该次查找没有找到符合要求的数据。

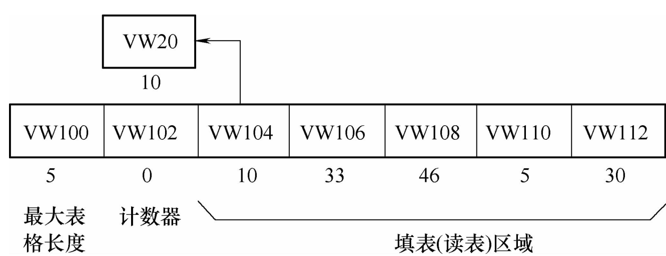


图 10-43　M0.1 接通第 5 次后

由于 AD _ T _ TBL 填表指令能够填写的表格长度范围是 1 ~ 100，对应数值所在的位置是 0 ~ 99。CMD 可能的数值是 1、2、3 和 4，分别代表 = 、 < > 、 < 和 > 。

图 10-44 所示的程序中，要求先使用 AD _ T _ TBL 填表指令填写好表格，然后使用 TBL _ FIND 查表指令在表格中查找符合要求的数据，MB10 显示符合要求的数据的总数。

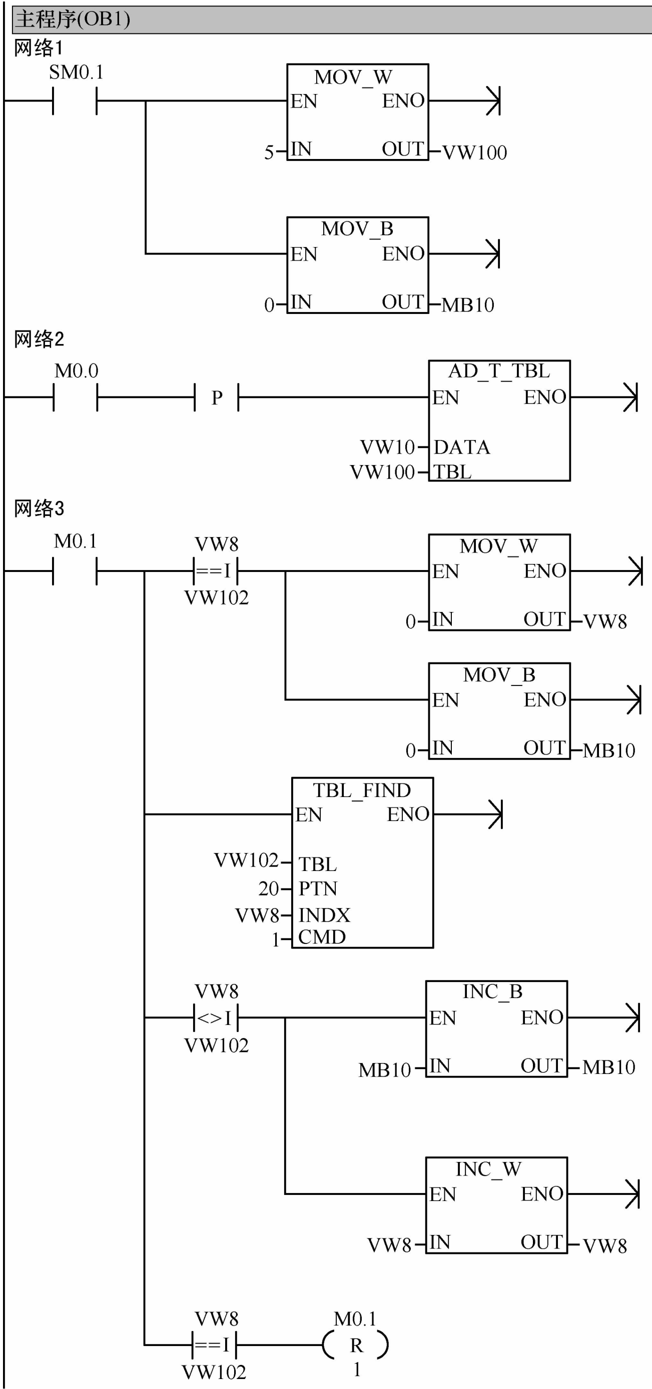


图 10-44　TBL _ FIND 指令应用

实际应用中，LIFO 后进先出指令、FIFO 先进先出指令和 AD _ T _ TBL 填表指令可以同时向相同或不同的表格读（填）表操作，只要保证表格不是空表（实际填表数不为零），LIFO 指令和 FIFO 指令操作有效；只要保证表格不是满表（实际填表数小于最大表格长度）AD _ T _ TBL 填表指令操作有效。

在使用 LIFO 指令、FIFO 指令和 AD _ T _ TBL 指令时，最好使用脉冲指令或其他确保填表指令的驱动信号只接通一个扫描周期的指令。

问 4：在应用表指令时需要注意些什么?

答：应用表指令需要注意如下两项：

一是间接寻址时注意不要超出寻址范围。

二是脉冲执行与连续执行是有区别的。

三是注意对空表进行读，对满表进行写。

第 11 章　与或异或逻辑指令

11.1　WAND _ B/W/DW

问：WADN 与逻辑指令基本功能是什么？ WADN 与逻辑指令样式是怎样的？ 如何应用 WADN 与逻辑指令？ 在应用 WADN 与逻辑指令时需要注意些什么？

问 1：WADN 与逻辑指令基本功能是什么？

答：WAND 与逻辑指令基本功能是对两个输入数值（IN1 和 IN2）的对应二进制位执行“与”（与也就是相乘）操作，并将结果放置在（OUT）中。如果结果为零，SM1.0 零标志位为“1”。

问 2：WADN 与逻辑指令样式是怎样的？

答：WAND 与逻辑指令按照操作数不同又分字节、字和双字，这些逻辑指令样式分别如图 11-1～图 11-3 所示。



图 11-1　WAND _ B 指令样式



图 11-2　WAND _ W 指令样式

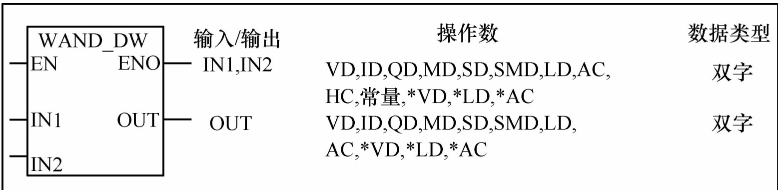


图 11-3　WAND _ DW 指令样式

问 3：如何应用 WADN 与逻辑指令？

答：WAND _ B 字节与指令对两个输入字节数值（IN1 和 IN2）的对应二进制位执行“与”操作，并将结果放置在（OUT）。如果结果为零，SM1.0 零标志位为“1”。

WAND _ W 字与指令对两个输入字数值（IN1 和 IN2）的对应二进制位执行“与”操作，并将结果放置在（OUT）。如果结果为零，SM1.0 零标志位为“1”。

WAND _ DW 双字与指令对两个输入双字数值（IN1 和 IN2）的对应二进制位执行“与”操作，并将结果放置在（OUT）。如果结果为零，SM1.0 零标志位会为“1”。

在图 11-4 所示的位与逻辑程序中，I0.0、I0.1、Q0.0 的状态关系（I0.0 ∧ I0.1 = Q0.0），见表 11-1 所示。

从表 11-1 可以看出，位与的输入状态和输出状态的关系表达为：有“0”出“0”，意思是说两个输入当中，其中有一个或以上的状态为“0”，输出就为“0”。

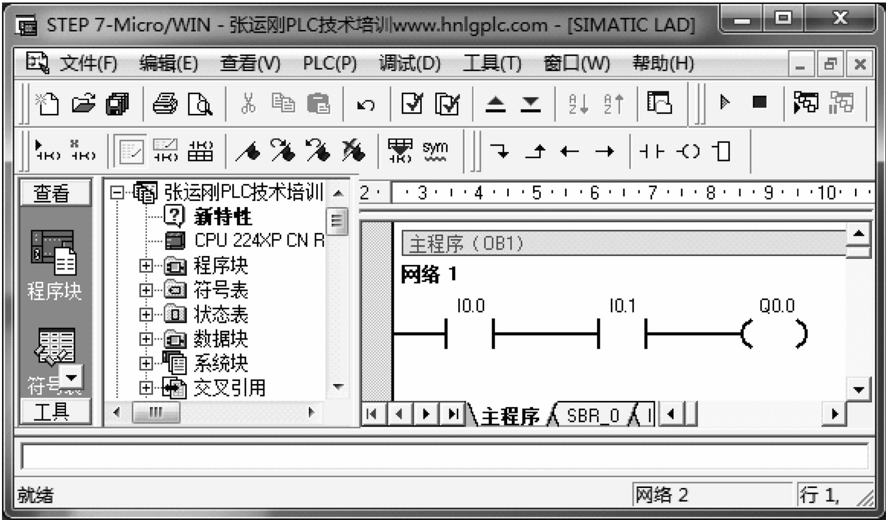


图 11-4　位与逻辑

表 11-1　位 与 逻 辑

I0.0	I0.1	Q0.0	I0.0	I0.1	Q0.0
0	0	0	0	1	0
1	0	0	1	1	1

在图 11-5 所示的字节与逻辑程序中，当 I0.0 接通，VB10 二进制数各位和 VB11 二进制数各位“与”，结果传送到 VB12。

比如：VB10 = 2#1100 1110，

VB11 = 2#1111 0000，

当 I0.0 接通，VB12 = 2#1100 0000。

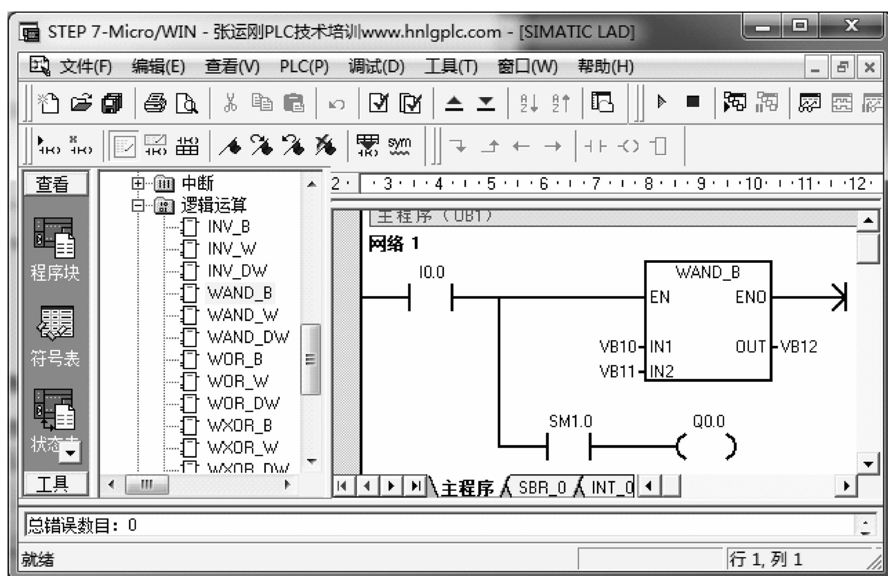


图 11-5　WAND _ B 指令应用

在图 11-6 所示的字与逻辑程序中，当 I0.0 接通，VW10 二进制数各位和 VW12 二进制数各位“与”，结果传送到 VW14。

比如：VW10 = 2#1100 1110 0110 0011，

VW12 = 2#1111 0000 1100 0111，

当 I0.0 接通，VW14 = 2#1100 0000 0100 0011。

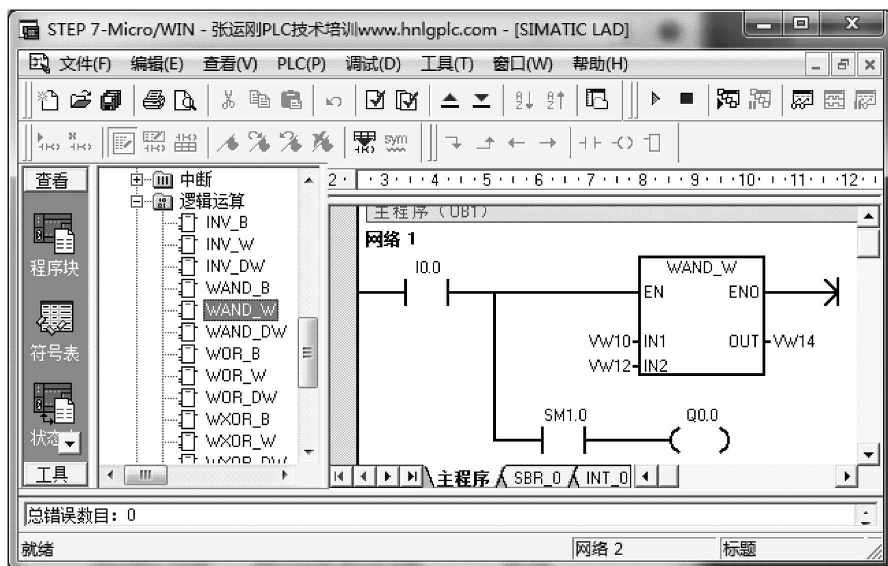


图 11-6　WAND _ W 指令应用

在图 11-7 所示的双字与逻辑程序中，当 I0.0 接通，VD10 二进制数各位和 VD14 二进制数各位“与”，结果传送到 VD18。

比如：VD10 = 2#1100 1110 0110 0011 1100 1110 0110 0011，
VD14 = 2#1111 0000 1100 0111 1111 0000 1100 0111，
当 I0.0 接通，VD18 = 2#1100 0000 0100 0011 1100 0000 0100 0011。

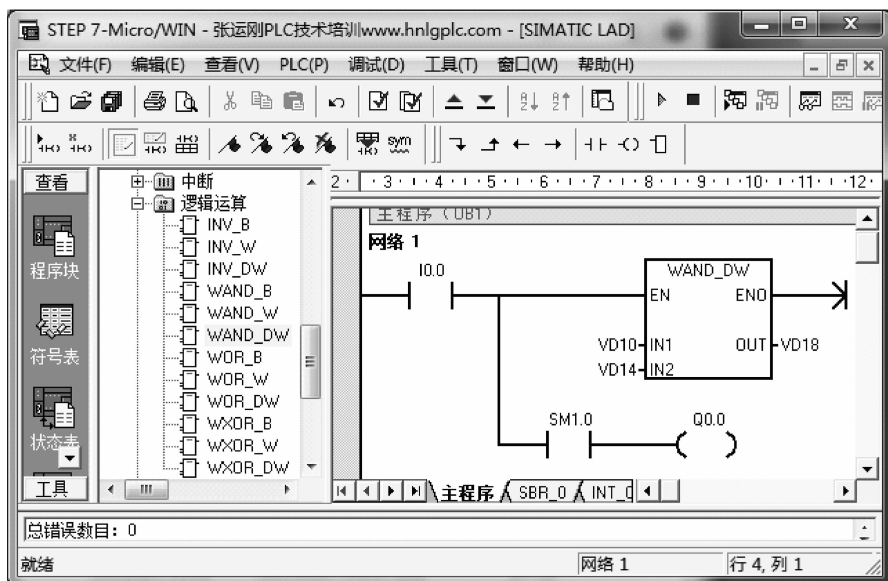


图 11-7　WAND _ DW 指令应用

在图 11-8 所示的程序中，当 I0.0 接通，VB10 为任意数值，Q0.4、Q0.5、Q0.6 和 Q0.7 都是没有输出的，像被屏蔽了一样。比如 VB10 = HFF，接通 I0.0 后，监控 Q0.0 ~ Q0.7，除了 Q0.4、Q0.5、Q0.6 和 Q0.7 的状态为“0”外，其他都为“1”。

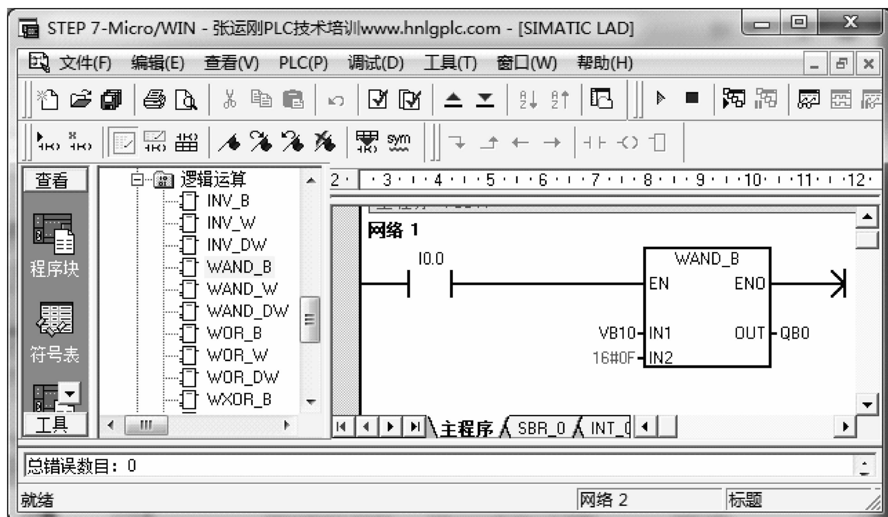


图 11-8　逻辑与指令应用

问 4：在应用 WADN 与逻辑指令时需要注意些什么？

答：在应用 WADN 与逻辑指令时需要注意以下两点：

一是间接寻址时注意不要超出寻址范围。

二是脉冲执行与连续执行是有区别的。

三是与逻辑可以把指定的位变为“0”。

11.2　WOR _ B/W/DW

问：WOR 或逻辑指令基本功能是什么?
WOR 或逻辑指令样式是怎样的?
如何应用或逻辑指令?
在应用 WOR 或逻辑指令时需要注意些什么?

问 1：WOR 或逻辑指令基本功能是什么?

答： WOR 或逻辑指令基本功能是对两个输入数值（IN1 和 IN2）的对应二进制位执行“或”（或也就是相加）操作，并将结果放置在（OUT）中。如果结果为零，SM1.0 零标志位为“1”。

问 2：WOR 或逻辑指令样式是怎样的?

答： WOR 或逻辑指令按照操作数不同又分字节、字和双字，这些逻辑指令样式分别如图 11-9 ~ 图 11-11 所示。



图 11-9　WOR _ B 指令样式



图 11-10　WOR _ W 指令样式



图 11-11　WOR _ DW 指令样式

问 3：如何应用或逻辑指令?

答： WOR _ B 字节或指令对两个输入字节数值（IN1 和 IN2）的对应二进制位执行

“或”操作，并将结果放置在（OUT）。如果结果为零，SM1.0 零标志位为“1”。

WOR _ W 字或指令对两个输入字数值（IN1 和 IN2）的对应二进制位执行“或”操作，并将结果放置在（OUT）。如果结果为零，SM1.0 零标志位为“1”。

WOR _ DW 双字或指令对两个输入双字数值（IN1 和 IN2）的对应二进制位执行“或”操作，并将结果放置在（OUT）。如果结果为零，SM1.0 零标志位会为“1”。

在图 11-12 所示的位或逻辑程序中，I0.0、I0.1、Q0.0 的状态关系（I0.0 ∨ I0.1 = Q0.0），见表 11-2 所示。

从表 11-2 可以看出，位或逻辑的输入状态和输出状态的关系表达为：有“1”出“1”，意思是说两个输入当中，其中有一个或以上的状态为“1”，输出就为“1”。

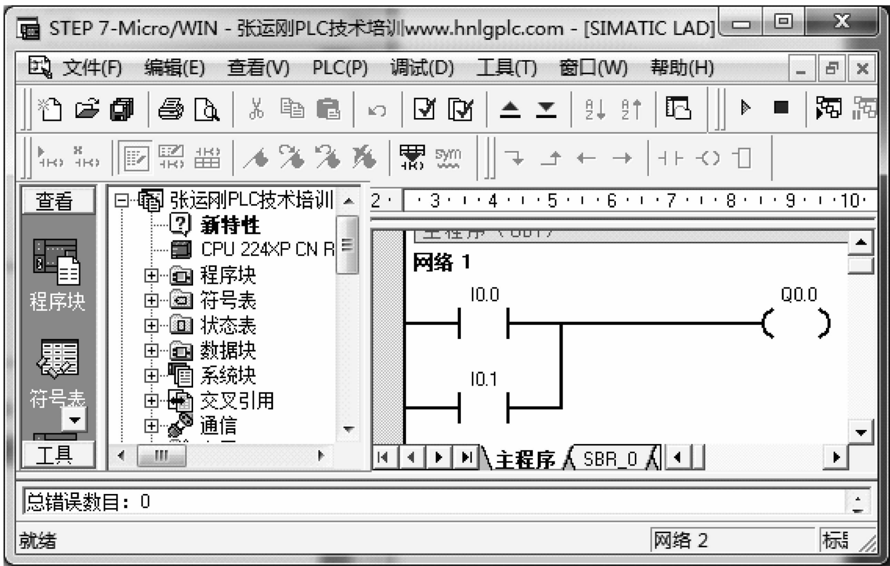


图 11-12　位或逻辑

表 11-2　位 或 逻 辑

I0.0	I0.1	Q0.0	I0.0	I0.1	Q0.0
0	0	0	0	1	1
1	0	1	1	1	1

在图 11-13 所示的字节或逻辑程序中，当 I0.0 接通，VB10 二进制数各位和 VB11 二进制数各位“或”，结果传送到 VB12。

比如：VB10 = 2#1100 1110，

VB11 = 2#1111 0000，

当 I0.0 接通，VB12 = 2#1111 1110。

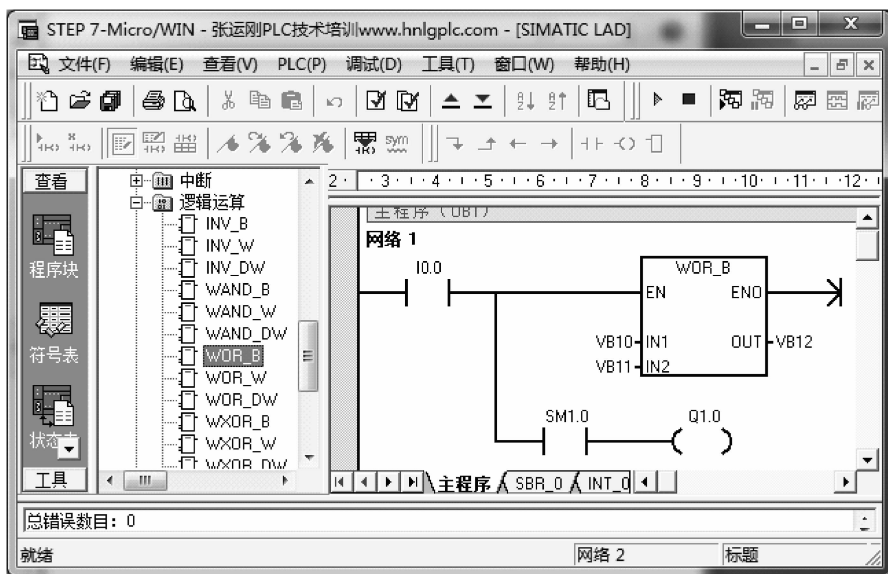


图 11-13　WOR _ B 指令应用

在图 11-14 所示的字或逻辑程序中，当 I0.0 接通，VW10 二进制数各位和 VW12 二进制数各位“或”，结果传送到 VW14。

比如：VW10 = 2#1100 1110 0110 0011，

VW12 = 2#1111 0000 1100 0111，

当 I0.0 接通，VW14 = 2#1111 1110 1110 0111。

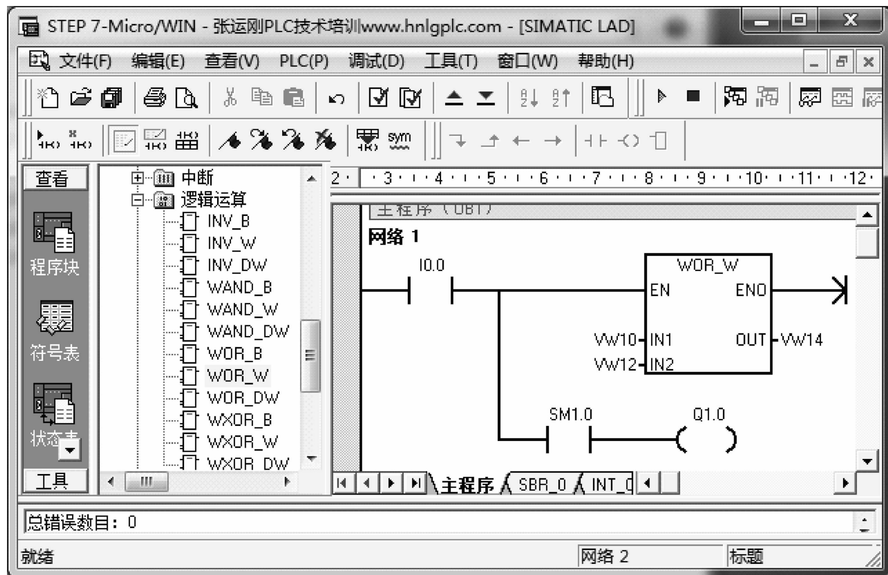


图 11-14　WOR _ W 指令应用

在图 11-15 所示的双字或逻辑程序中，当 I0.0 接通，VD10 二进制数各位和 VD14 二进制数各位“或”，结果传送到 VD18。

比如：VD10 = 2#1100 1110 0110 0011 1100 1110 0110 0011，
　　　VD14 = 2#1111 0000 1100 0111 1111 0000 1100 0111，
当 I0.0 接通，VD18 = 2#1111 1110 1110 0111 1111 1110 1110 0111。

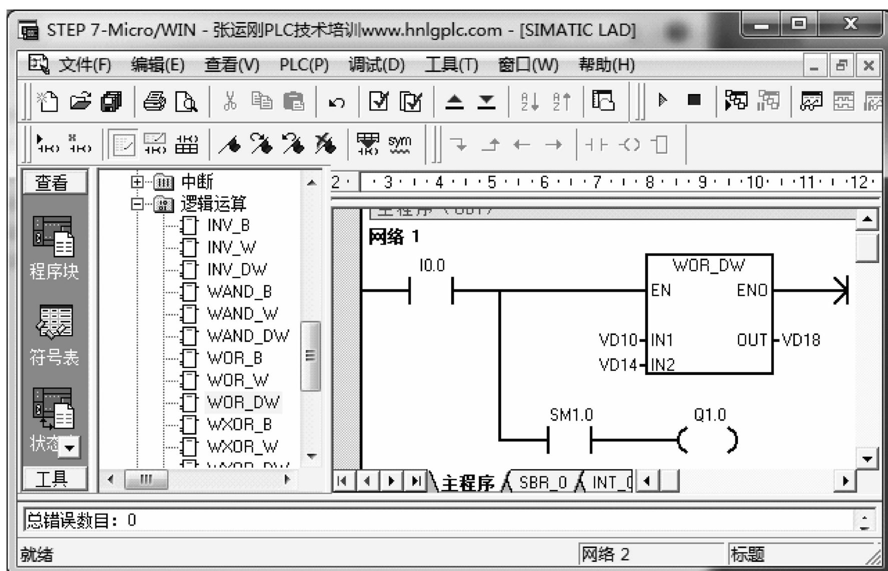


图 11-15　WOR _ DW 指令应用

在图 11-16 所示的字节或逻辑应用程序中，当 I0.0 接通，VB10 为任意数值，Q0.4、Q0.5、Q0.6 和 Q0.7 都是有输出的，像被钳位为“1”了一样。比如 VB10 = H00，接通 I0.0，监控 Q0.0 ~ Q0.7，除了 Q0.4、Q0.5、Q0.6 和 Q0.7 的状态为“1”外，其他都为“0”。

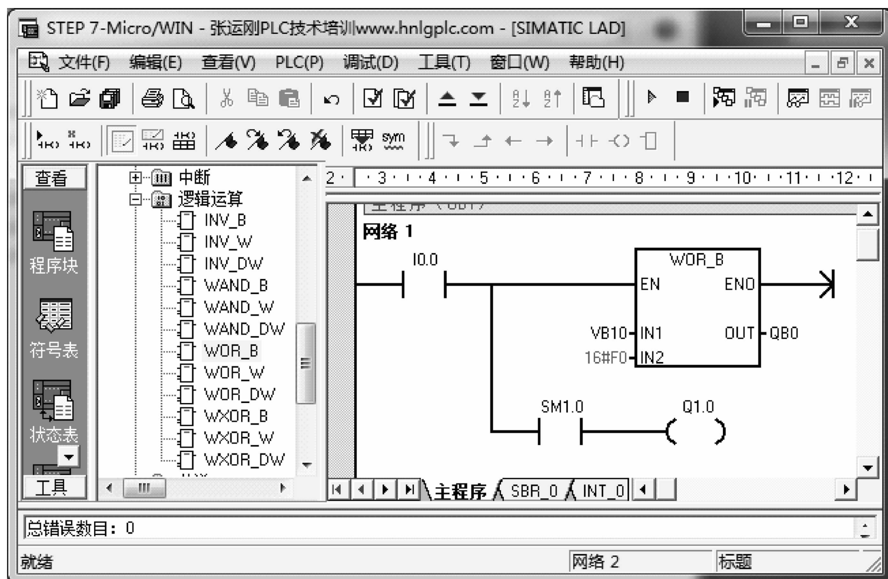


图 11-16　逻辑或指令应用

问 4：在应用 WOR 或逻辑指令时需要注意些什么？

答：在应用 WOR 或逻辑指令时需要注意以下两点：

一是间接寻址时注意不要超出寻址范围。

二是脉冲执行与连续执行是有区别的。

三是或逻辑可以把指定的位变为“1”。

11.3 WXOR _ B/W/DW

> 问：WXOR 异或逻辑指令基本功能是什么？
> WXOR 异或逻辑指令样式是怎样的？
> 如何应用 WXOR 异或逻辑指令？
> 在应用 WXOR 异或逻辑指令时需要注意些什么？

问 1：WXOR 异或逻辑指令基本功能是什么？

答：WXOR 异或逻辑指令基本功能是对两个输入数值（IN1 和 IN2）的对应二进制位执行“异或”操作，并将结果放置在（OUT）中。如果结果为零，SM1.0 零标志位为“1”。

问 2：WXOR 异或逻辑指令样式是怎样的？

答：WXOR 异或逻辑指令按照操作数不同又分字节、字和双字，这些字逻辑指令样式分别如图 11-17 ~ 图 11-19 所示。



图 11-17　WXOR _ B 指令样式



图 11-18　WXOR _ W 指令样式



图 11-19　WXOR _ DW 指令样式

问 3：如何应用 WXOR 异或逻辑指令？

答：WXOR _ B 字节异或指令对两个输入字节数值（IN1 和 IN2）的对应二进制位执行

“异或”操作，并将结果放置在（OUT）。如果结果为零，SM1.0 零标志位为“1”。

WXOR _ W 字异或指令对两个输入字数值（IN1 和 IN2）的对应二进制位执行“异或”操作，并将结果放置在（OUT）。如果结果为零，SM1.0 零标志位为“1”。

WXOR _ DW 双字异或指令对两个输入双字数值（IN1 和 IN2）的对应二进制位执行“异或”操作，并将结果放置在（OUT）。如果结果为零，SM1.0 零标志位会为“1”。

在图 11-20 所示的位异或逻辑程序中，I0.0、I0.1、Q0.0 的状态关系（I0.0 ∀ I0.1 = Q0.0），见表 11-3。

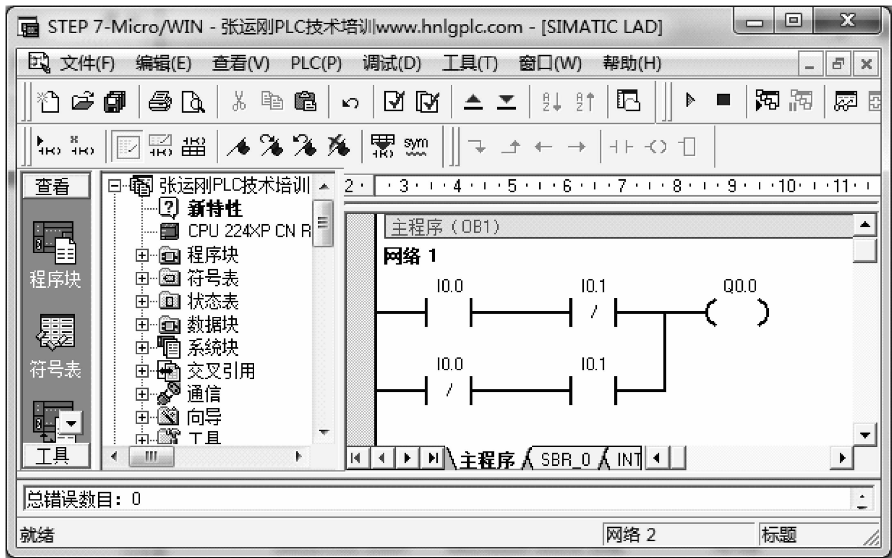


图 11-20　位异或逻辑

表 11-3　位异或逻辑

I0.0	I0.1	Q0.0	I0.0	I0.1	Q0.0
0	0	0	0	1	1
1	0	1	1	1	0

从表 11-3 可以看出，位异或的输入状态和输出状态的关系表达为：相同出“0”，意思是说两个输入当中，其中两个的状态都为“0”或两个的状态都为“1”，输出就为“0”。

在图 11-21 所示的字节异或逻辑程序中，当 I0.0 接通，VB10 二进制数各位和 VB11 二进制数各位“异或”，结果传送到 VB12。

比如：VB10 = 2#1100 1110，

VB11 = 2#1111 0000，

当 I0.0 接通，VB12 = 2#0011 1110。

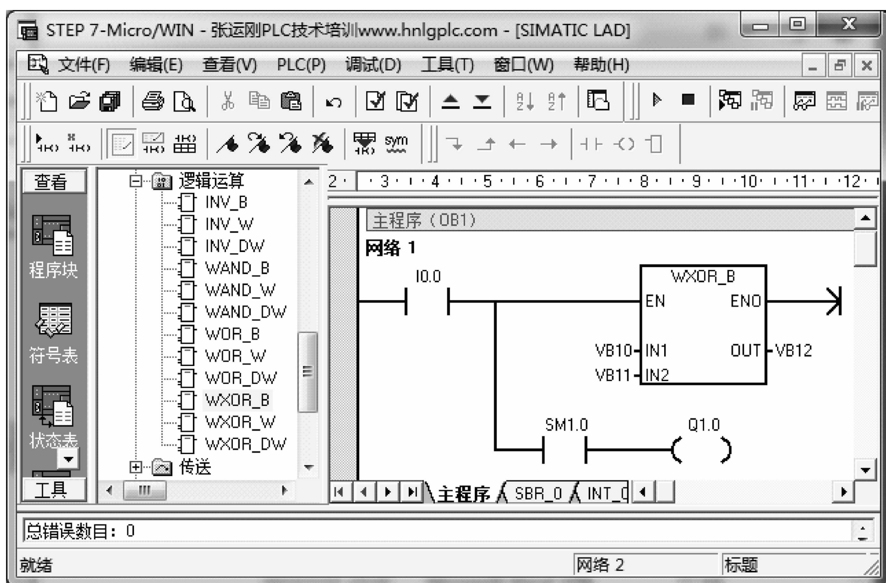


图 11-21　WXOR _ B 指令应用

在图 11-22 所示的字异或逻辑程序中，当 I0.0 接通，VW10 二进制数各位和 VW12 二进制数各位“异或”，结果传送到 VW14。

比如：VW10 = 2#1100 1110 0110 0011，

VW12 = 2#1111 0000 1100 0111，

当 I0.0 接通，VW14 = 2#0011 1110 1010 0100。

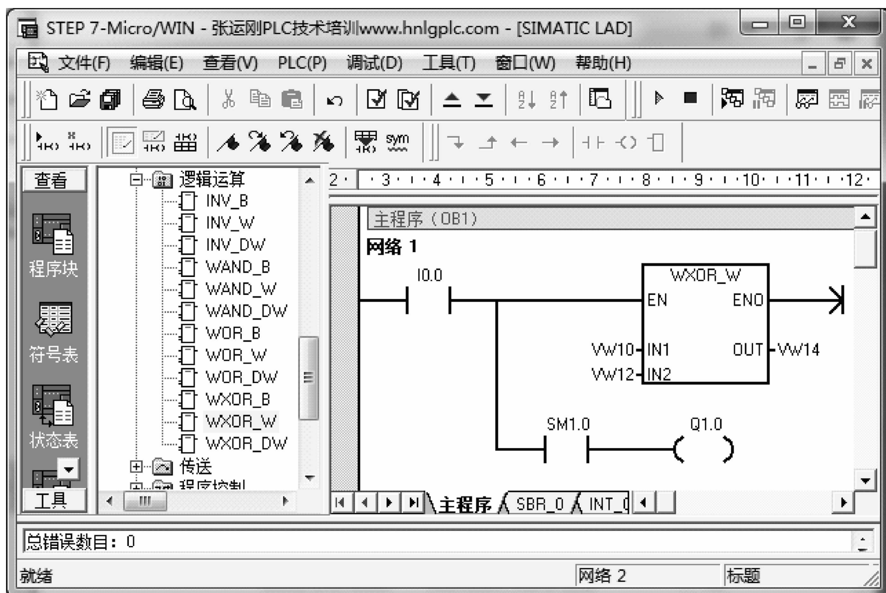


图 11-22　WXOR _ W 指令应用

在图 11-23 所示的双字异或逻辑程序中，当 I0.0 接通，VD10 二进制数各位和 VD14 二进制数各位“异或”，结果传送到 VD18。

比如：VD10 = 2#1100 1110 0110 0011 1100 1110 0110 0011，

VD14 = 2#1111 0000 1100 0111 1111 0000 1100 0111，

当 I0.0 接通，VD18 = 2#0011 1110 1010 0100 0011 1110 1010 0100。

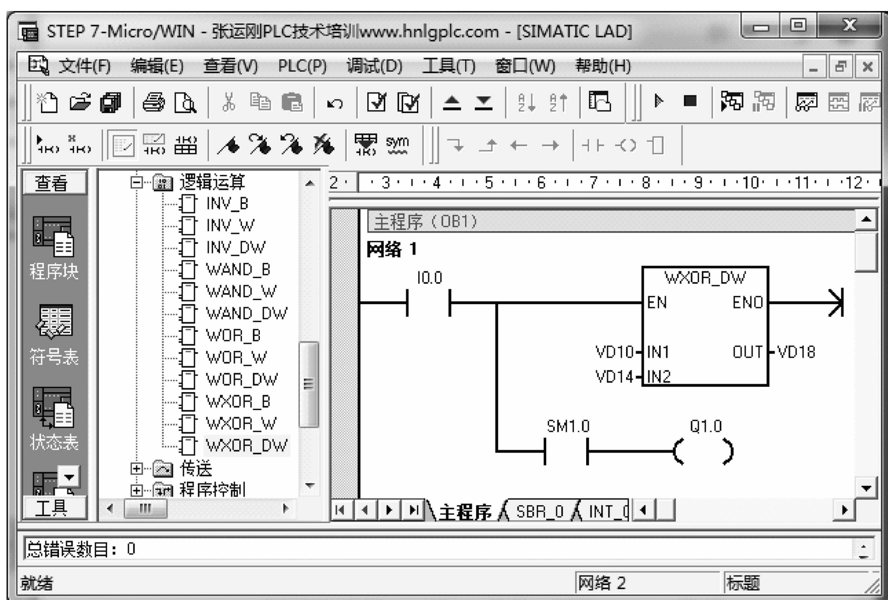


图 11-23　WXOR _ DW

在图 11-24 所示的异或逻辑程序中，SM0.5 是秒脉冲信号，将 IB0 ~ IB3 输入检测信号分成 4 组，4s 为一个检测周期，在一个检测周期内每秒对每一组进行检测：

当 SM0.5 第一次接通，VD100 指向 IB0，检测 IB0（即 I0.0 ~ I0.7 八位）；

当 SM0.5 第二次接通，VD100 指向 IB1，检测 IB1（即 I1.0 ~ I1.7 八位）；

当 SM0.5 第三次接通，VD100 指向 IB2，检测 IB2（即 I2.0 ~ I2.7 八位）；

当 SM0.5 第四次接通，VD100 指向 IB3，检测 IB3（即 I3.0 ~ I3.7 八位）；

当 SM0.5 第一次接通，VD100 指向 IB0，检测 IB0（即 I0.0 ~ I0.7 八位）。

每组检测都与“0”异或运算，如果该组有一个或以上输入点导通，异或结果（VB10）不为零，M0.0 就有报警输出，这时如果没有对报警进行确认复位（接通 M0.1），系统将不进行扫描检测。

问 4：在应用 WXOR 异或逻辑指令时需要注意些什么？

答：在应用 WXOR 异或逻辑指令时需要注意以下几点：

一是间接寻址时注意不要超出寻址范围。

二是脉冲执行与连续执行是有区别的。

三是异或逻辑可以把指定的位实现取反变换。

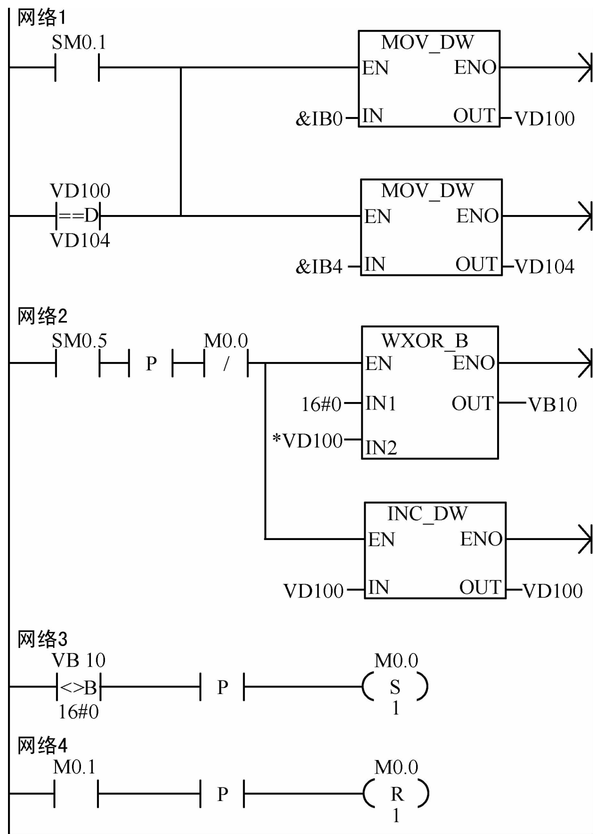


图 11-24　字异或逻辑指令应用

第 12 章　程序控制指令

12.1　JMP/LBL

问：跳转指令基本动作是什么？ 跳转指令样式是怎样的？ 如何应用跳转指令？ 在应用跳转指令时需要注意些什么？

问 1：跳转指令基本动作是什么？

答：跳转指令需要使用（JMP n）和（LBL n）配合，是控制程序指针跳转到指定标签（n）处执行程序，标签（n）的数值范围是 0～255。

问 2：跳转指令样式是怎样的？

答：跳转指令样式如图 12-1 所示。

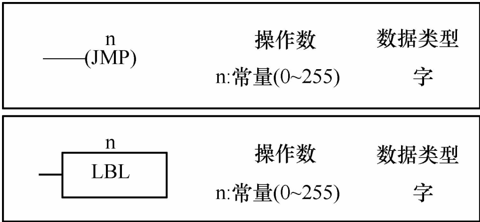


图 12-1　跳转指令样式

问 3：如何应用跳转指令？

答：跳转指令两个标签必须在同一个程序块内，这些程序可以是主程序，也可以是子程序或者是中断程序。在程序中有步进阶梯指令时，可以在 SCR 段中使用“跳转”指令，但对应的“标签”指令必须位于相同的 SCR 段内。

在图 12-2 所示的程序中，程序执行顺序是：

当 I0.2 =0，程序执行顺序是：网络 1→网络 2→网络 3→网络 4→网络 1；

当 I0.2 =1，程序执行顺序：网络 1→网络 3→网络 4→网络 1。

在图 12-3 所示的程序中，当 I0.0 =0 和 I0.1 =1 时，Q0.1、I15.0、M0.0 和 V0.0 就接通。

当 Q0.1、I15.0、M0.0 和 V0.0 在接通状态，如果这时 I0.0 =1，不管 I0.1 接通还是断开，Q0.1、I15.0、M0.0 和 V0.0 的状态将保持接通。

当 Q0.1、I15.0、M0.0 和 V0.0 在断开状态，如果这时 I0.0 =1，不管 I0.1 接通还是断开，Q0.1、I15.0、M0.0 和 V0.0 的状态将保持断开。

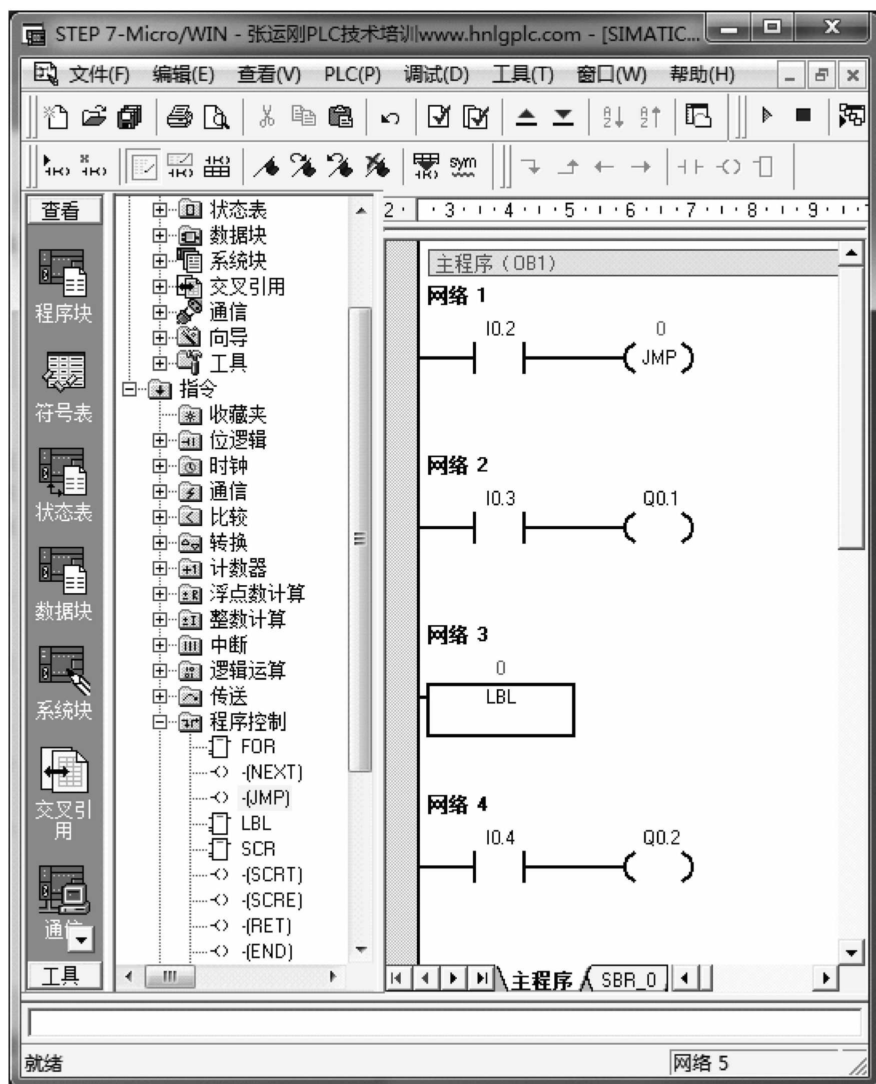


图 12-2　跳转指令控制流程

在图 12-4 所示的程序中：

当使 I0.0 =0 和 I0.1 =1 时，T32、T33、T37 开始计时，这时如果 I0.0 先接通，不管 I0.1 是接通还是断开程序状态将是：

代表 1ms 的定时器 T32 和代表 10ms 的定时器 T33 继续计时，当定时器的经过值等于设定值时，其常开触点会接通；代表 100ms 的定时器 T37 则暂停计时，其经过值保持不变。

当使 I0.1 =0 和 I0.0 =1，最后使 I0.1 =1 时，T32、T33、T37 都不计时。

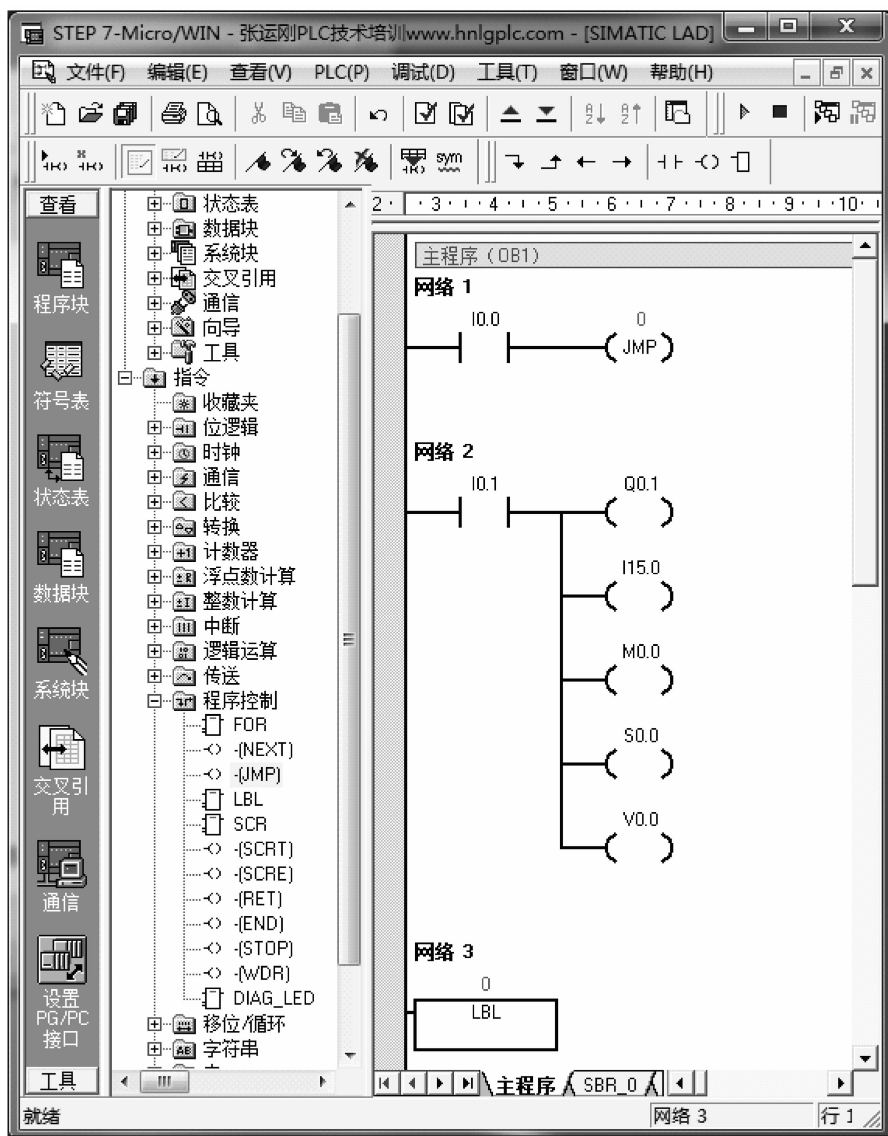


图 12-3　I、Q、M、S、V 在跳转指令中的状态

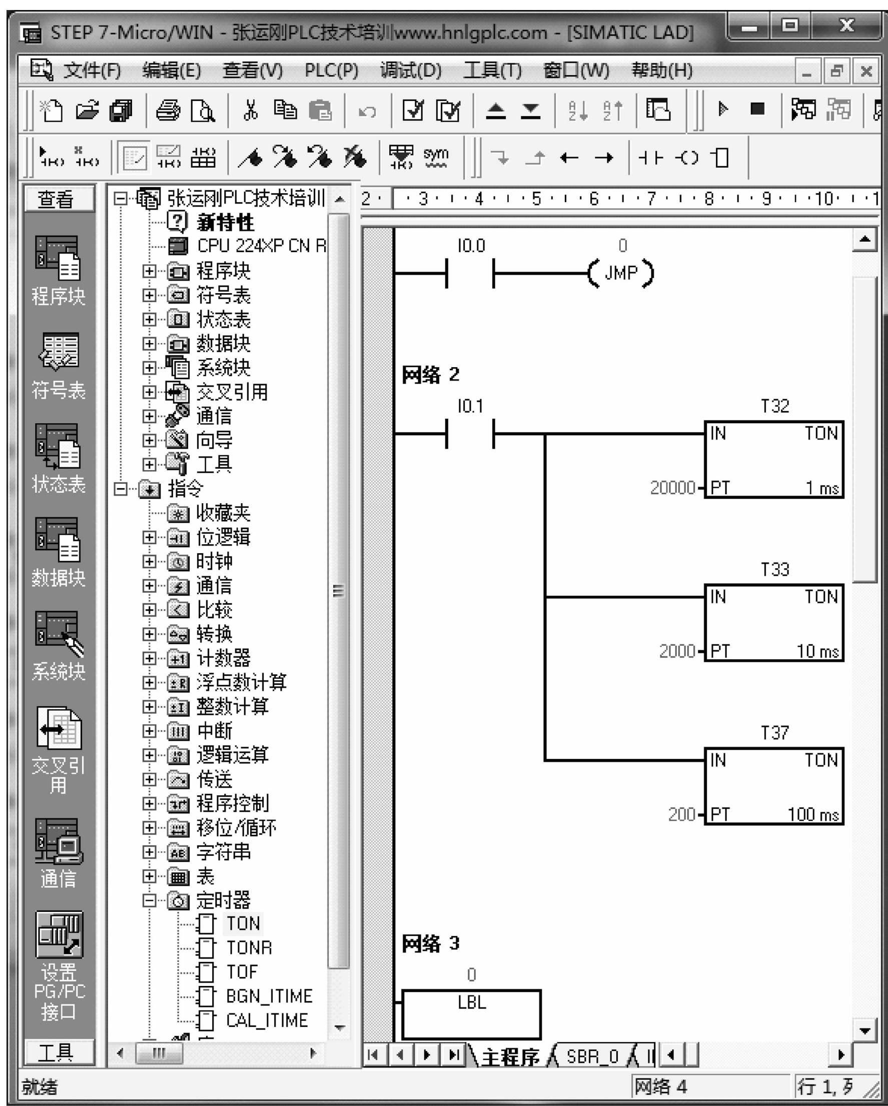


图 12-4　定时器在跳转指令中的状态

在图 12-5 所示程序中，输入与输出的关系规律见表 12-1。

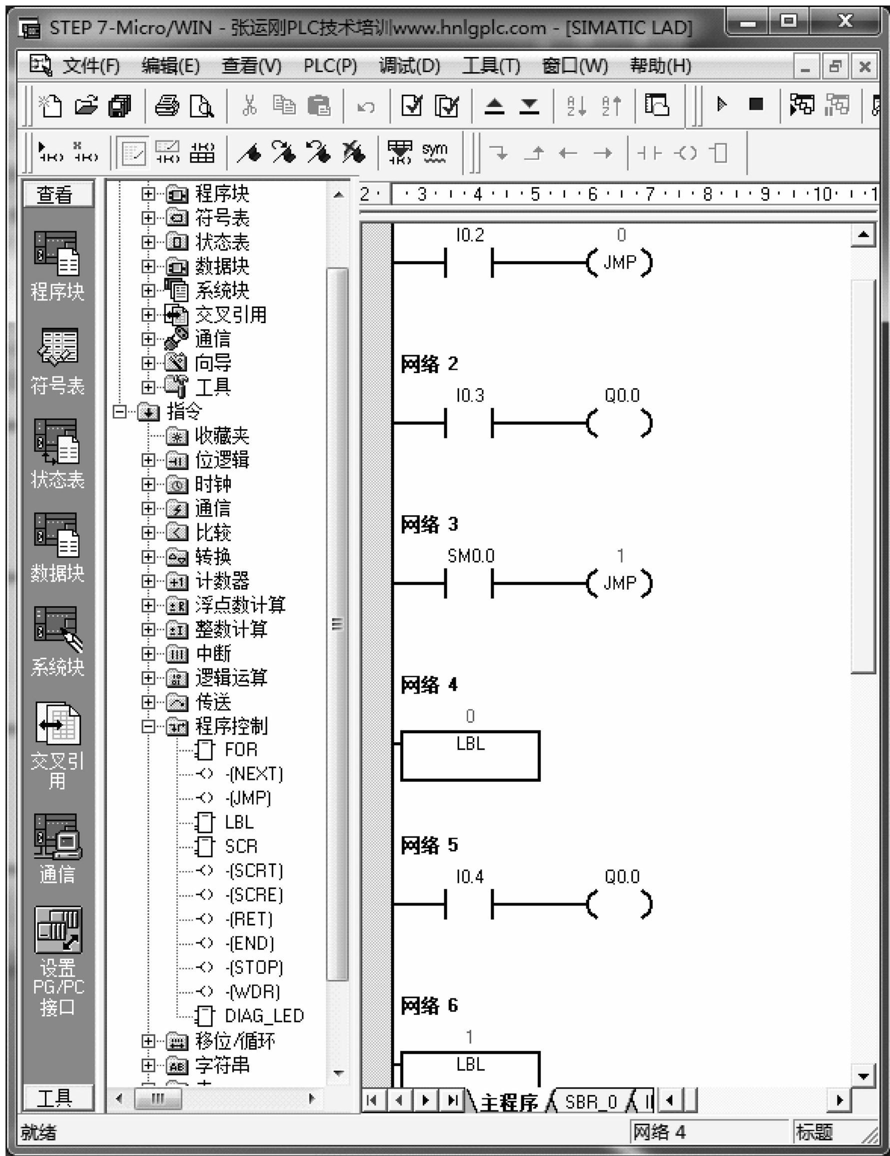


图 12-5　跳转指令中的双重线圈动作

表 12-1　跳转指令中的双重线圈动作

I0. 2	I0. 3	I0. 4	Q0. 0	I0. 2	I0. 3	I0. 4	Q0. 0
1	0	0	0	0	0	0	0
	0	1	1		0	1	0
	1	0	0		1	0	1
	1	1	1		1	1	1

在图 12-6 所示程序中：

当 I0. 0 = 1 时，程序执行流程：网络 1→网络 6→网络 7→网络 1；

当 I0. 0 = 0、I0. 2 = 1 时，程序执行流程：网络 1→网络 2→网络 3→网络 6→网络 7→网络 1；

当 I0. 0 = 0、I0. 2 = 0 时，程序执行流程：网络 1→网络 2→网络 3→网络 4→网络 5→网络 1。

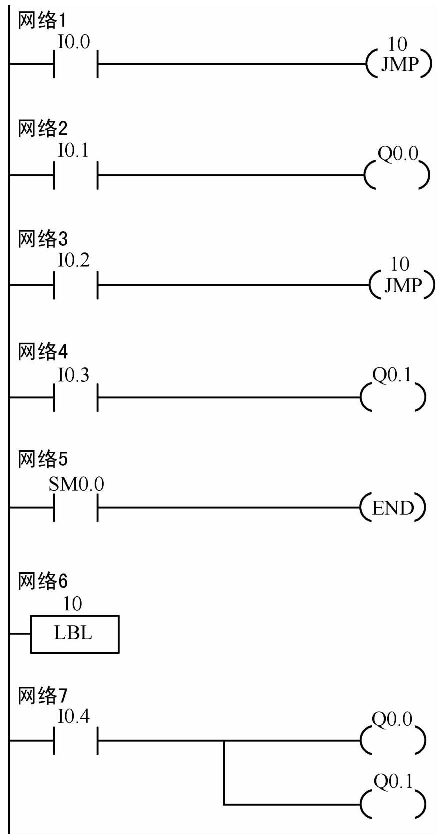


图 12-6　多处跳转到相同一个标签处

在实际工程中，常用跳转指令来实现手动挡与自动挡之间的切换。例如电动机的星—三角启动/停止控制系统，I/O 分配见表 12-2 所示。星—三角启动/停止控制系统控制程序如图 12-7 所示。

表 12-2　星—三角启动/停止控制系统 I/O 分配表

挡位	挡位选择	启动	星转三角	停止	星形输出	三角形输出	主输出
手动	I0. 0 = 1	0. 1	I0. 3	I0. 2	Q0. 0	Q0. 1	Q0. 2
自动	I0. 0 = 0		T37				

问 4：在应用跳转指令时需要注意些什么？

答： 应用跳转指令需要注意以下几项：

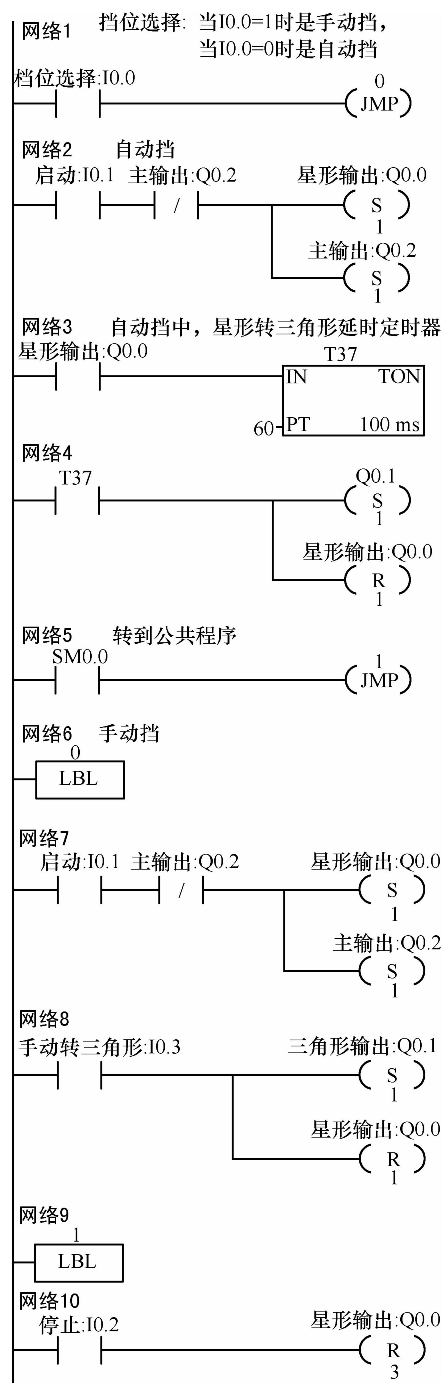


图 12-7　跳转指令典型应用例

一是当标号出现在跳转指令之前时，注意程序进入死循环造成看门狗超时错误。

二是当标号与跳转指令出现在不同程序块内，将会出现下载编译错误。

三是当跳转指令与标号两个数值不同，将会出现下载编译错误。

12. 2　ROR/NEXT

> 问：FOR _ NEXT 指令基本动作是什么？
> FOR _ NEXT 指令样式是怎样的？
> 如何应用 FOR _ NEXT 指令？
> 在应用 FOR _ NEXT 指令时需要注意些什么？

问 1：FOR _ NEXT 指令基本动作是什么？

答：FOR _ NEXT 指令基本动作是控制执行 FOR 和 NEXT 之间的指令次数的操作。使用 FOR _ NEXT 指令必须指定当前循环计数（INDX）、起始值（INIT）和结束值（FINAL）。"NEXT" 指令是 FOR 循环结束标记，并将堆栈顶值设为 1。每条 FOR 指令要求有一个 NEXT 指令，也就是说 FOR 指令和 NEXT 指令必须成对使用。

如果启用 FOR _ NEXT 指令时，它将初始值复制至循环计数（INDX），然后将继续循环执行程序，直至结束操作，除非从循环内部改变结束值，比如可以在 FOR _ NEXT 指令处于循环过程中改变结束值（FINAL）。

假定起始值（INIT）等于 1，结束值（FINAL）等于 10，FOR 与 NEXT 之间的指令被执行 10 次，INDX 值递增 "1"：1、2、3、…、10。如果起始值大于结束值，则不执行循环。每次执行 FOR 和 NEXT 之间的指令后，INDX 值递增 "1"，并将结果与结束值比较，如果循环计数（INDX）大于结束值，则终止循环。

问 2：FOR _ NEXT 指令样式是怎样的？

答：FOR _ NEXT 指令样式如图 12-8 所示。



图 12-8　FOR _ NEXT 指令样式

问 3：如何应用 FOR _ NEXT 指令？

答：在图 12-9 的 FOR _ NEXT 指令应用程序中，每当 I0. 2 从 0 变 1 时，VW1000 的数值会从 0 开始加到 50。

FOR _ NEXT 指令可以嵌套使用【在 FOR _ NEXT 指令中放置另外一个 FOR _ NEXT 指令】，嵌套深度可达 8 级。嵌套使用【在 FOR _ NEXT 指令如图 12-10 所示。

在图 12-10 所示的程序中，每当 I0. 2 从 0 变 1 一次，执行 INC _ B VB200 的指令 5 次，

因此 VB200 的数值等于 5；执行 INC _ B VB201 的指令 40 次，因此 VB201 的数值等于 40。

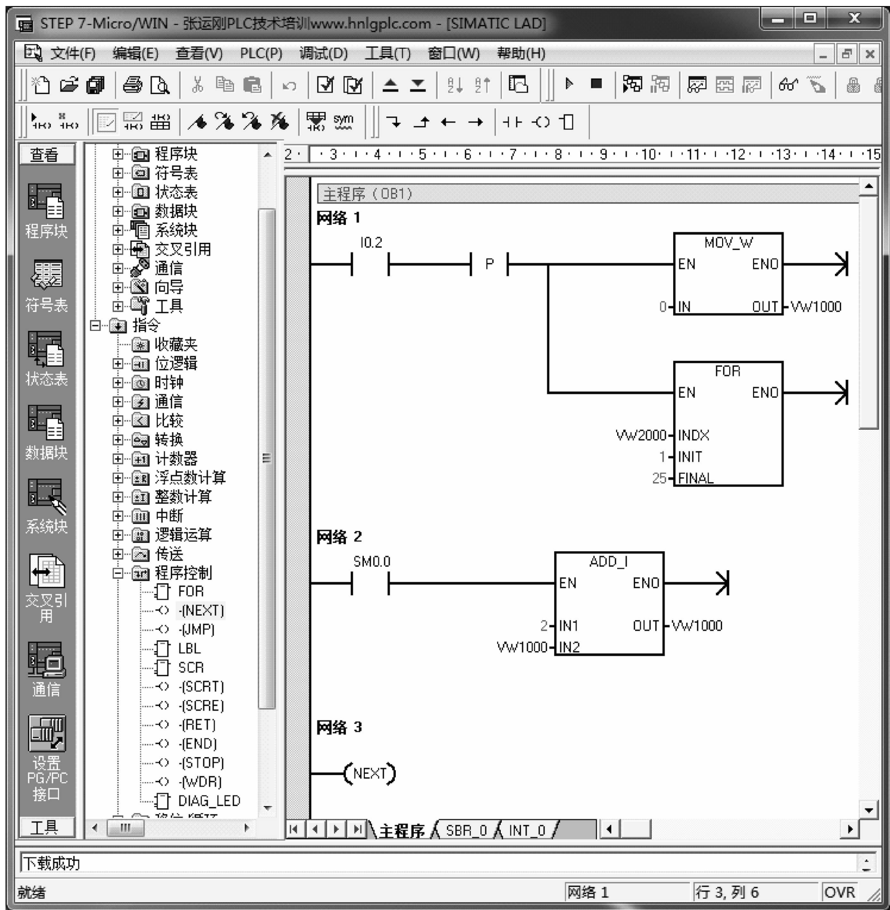


图 12-9　FOR _ NEXT 指令应用

问 4：在应用 FOR _ NEXT 指令时需要注意些什么？

答：应用 FOR _ NEXT 指令需要注意以下几项：

一是当要求循环次数太多，注意程序进入死循环造成看门狗超时错误。

二是当 FOR 与 NEXT 指令出现在不同程序块内，将会出现下载编译错误。

三是当嵌套使用时，FOR 与 NEXT 指令没有成对出现，将会出现下载编译错误。

四是间接寻址时，注意超出软元件的访问范围。

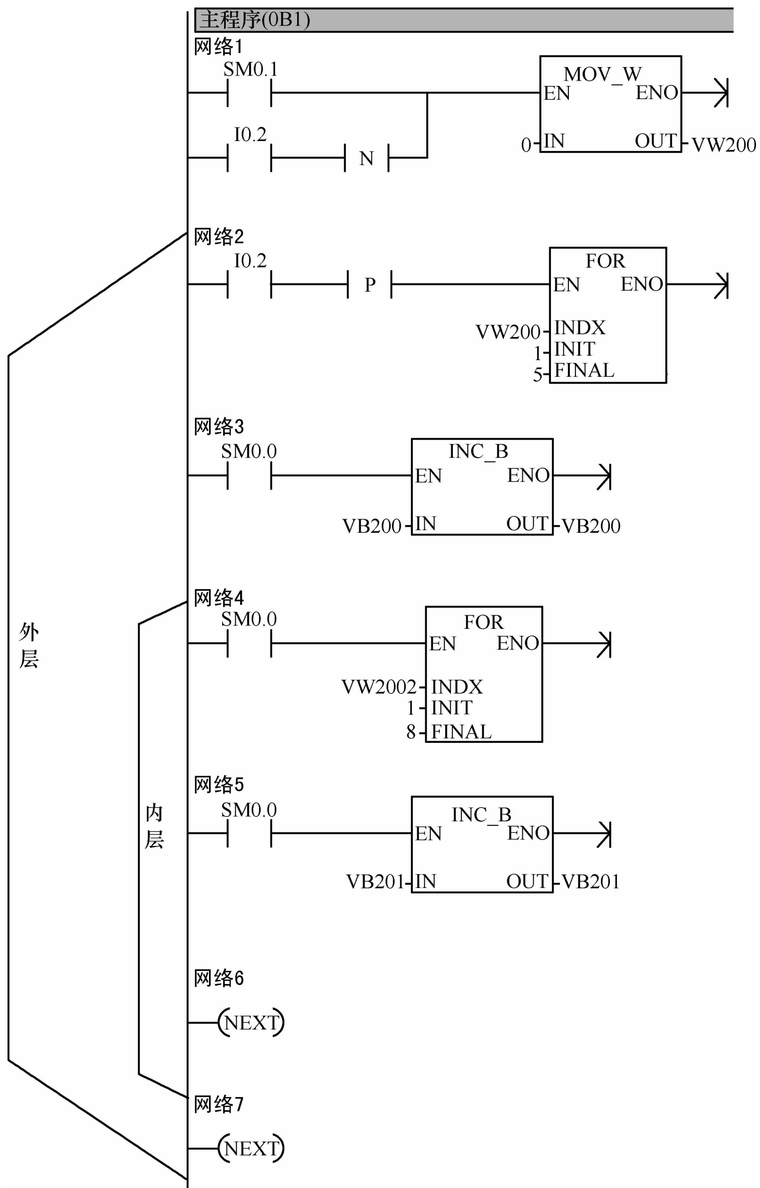


图 12-10　嵌套使用

12.3　END/STOP/WDR

问：END 指令基本动作是什么？
　　如何应用 END 指令？
　　在应用 END 指令时需要注意些什么？
　　STOP 指令基本动作是什么？
　　如何应用 STOP 指令？

在应用 STOP 指令时需要注意些什么? WDR 指令基本动作是什么? 如何应用 WDR 指令? 在应用 WDR 指令时需要注意些什么?

问 1：END 指令基本动作是什么?

答：END 指令是有条件结束指令，当执行了 END 指令将表示主程序结束。Micro/WIN 编程软件的编译器在编译成功后，自动在主程序中添加无条件结束指令。

问 2：如何应用 END 指令?

答：在图 12-11 所示的程序中，当 I0. 2 =0 时，接通 I0. 0，Q0. 0 和 Q0. 1 都会接通；当 I0. 2 =1、Q0. 1 =0 时，接通 I0. 0，只有 Q0. 0 接通而 Q0. 1 不会接通。但当 I0. 1 =1 时再接通 I0. 2，不管 I0. 0 是接通还是断开，Q0. 1 保持接通。

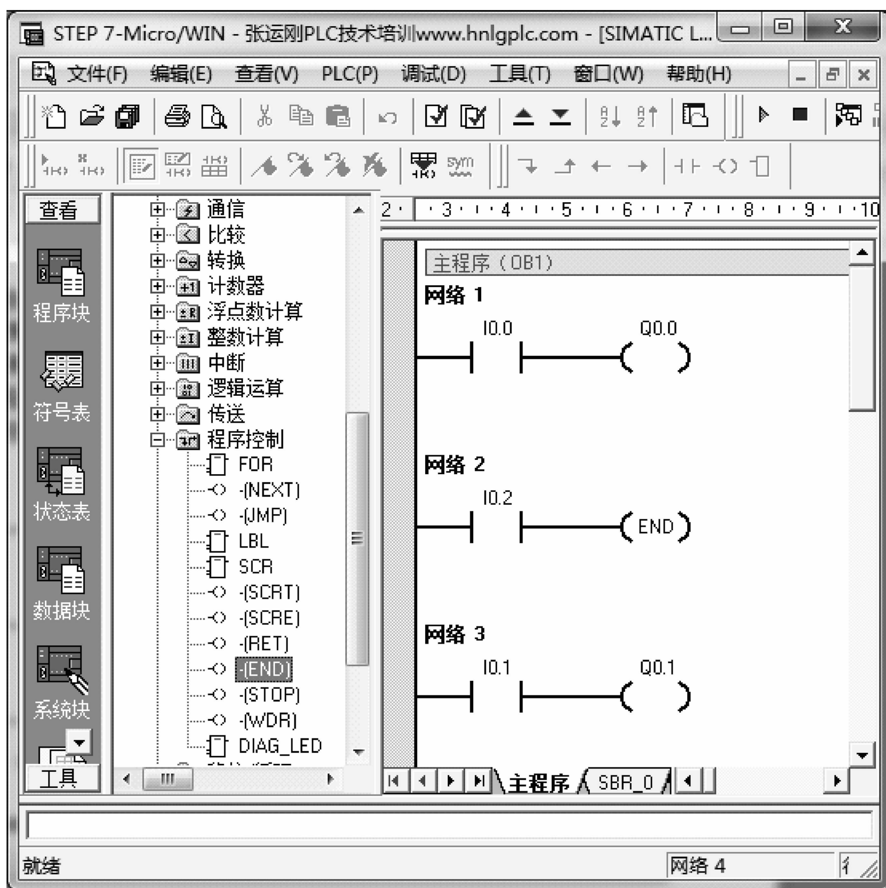


图 12-11　END 指令应用

问 3：在应用 END 指令时需要注意些什么?

答：当使用 END 指令时，需要注意：

只能在主程序（OB1）中使用，不能在子程序或者中断程序中使用。

问 4：STOP 指令基本动作是什么？

答：STOP 指令基本动作是，执行 STOP（停止）指令将会把 S7-200 CPU 的工作模式从 RUN（运行）转换为 STOP（停止）模式，终止程序执行。

问 5：如何应用 STOP 指令？

答：在图 12-12 所示的程序中，当 I0.2 =1 时，S7-200 CPU 的工作模式从 RUN（运行）转换为 STOP（停止）模式，终止程序执行。

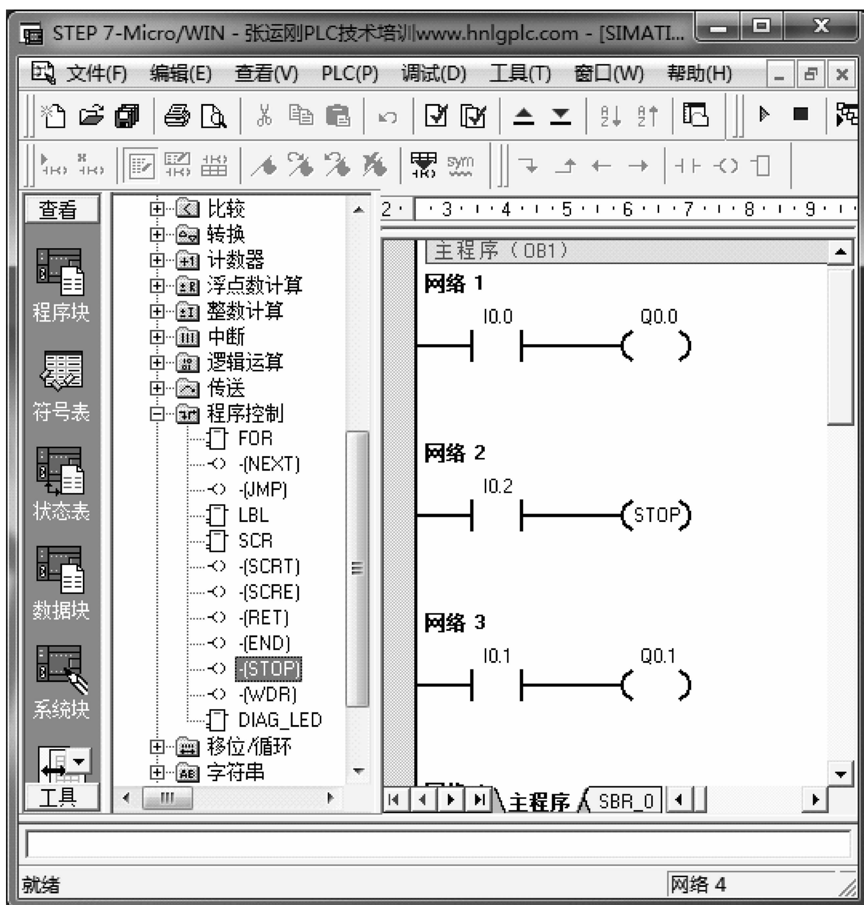


图 12-12　STOP 指令应用

在实际应用项目中，使用 STOP 指令目的是避免把故障危害扩大化而自动终止执行程序，是一种自保护措施。当 CPU 切换到停止模式后，无法使用程序让 CPU 恢复运行模式，只能依靠人工操作，让 CPU 恢复运行。

问 6：在应用 STOP 指令时需要注意些什么？

答：如果在中断程序中执行 STOP（停止）指令，中断程序立即终止，并忽略全部待执行的中断，转到扫描循环中的剩余程序，包括执行主程序和子程序，在当前扫描结束时从 RUN（运行）转换至 STOP（停止）模式。

问 7：WDR 指令基本动作是什么？

答：WDR 指令基本动作是重新触发 S7-200 CPU 的系统监视程序定时器，通俗说法是复

位看门狗定时器，每执行一次 WDR 指令就复位一次。执行 WDR 指令目的是延长扫描允许使用的时间，而不会出现监视超时错误。

如果使用监视程序定时器复位指令来允许执行要求很长扫描时间的程序，人为地将模式开关切换到 STOP（停止）位置，S7-200CPU 会在 1.4s 内转换为 STOP（停止）模式。

使用监视程序定时器复位指令时应当小心。如果使用循环指令阻止扫描完成或严重延迟扫描完成，下列程序只有在扫描循环完成后才能执行：

①通信（自由端口模式除外）；

②I/O 更新（立即 I/O 除外）；

③强制更新；

④SM 位更新（不更新 SMB0、SMB5 ~ SMB29）；

⑤运行时间诊断程序；

⑥10ms 和 100ms 计时器对于超过 25s 的扫描不能正确地累计时间；

⑦用于中断程序时的 STOP（停止）指令；

⑧配备离散输出的扩充模块还包括监视程序定时器，如果模块未被 S7-200 写入，监视程序定时器会关闭输出。对每个配备离散输出的扩充模块使用立即写入，在扩展扫描时间期间使正确的输出保持打开。

问 8：如何应用 WDR 指令?

答：在图 12-13 所示的程序中，当 M0.0 = 0，I0.0 接通，CPU 马上出现监视程序错误；当 M0.0 = 1 时，I0.0 接通，CPU 不会出现监视程序错误。查看 CPU 信息如图 12-14 所示，当 M0.0 = 1 时，I0.0 接通，程序扫描时间是 14883ms。

问 9：在应用 WDR 指令时需要注意些什么?

答：在应用 WDR 指令时需要注意的是不要滥用 WDR 指令。其实，在完成控制任务的提前，扫描周期时间越短越好。如果为了让 CPU 超时工作而不报警停机，

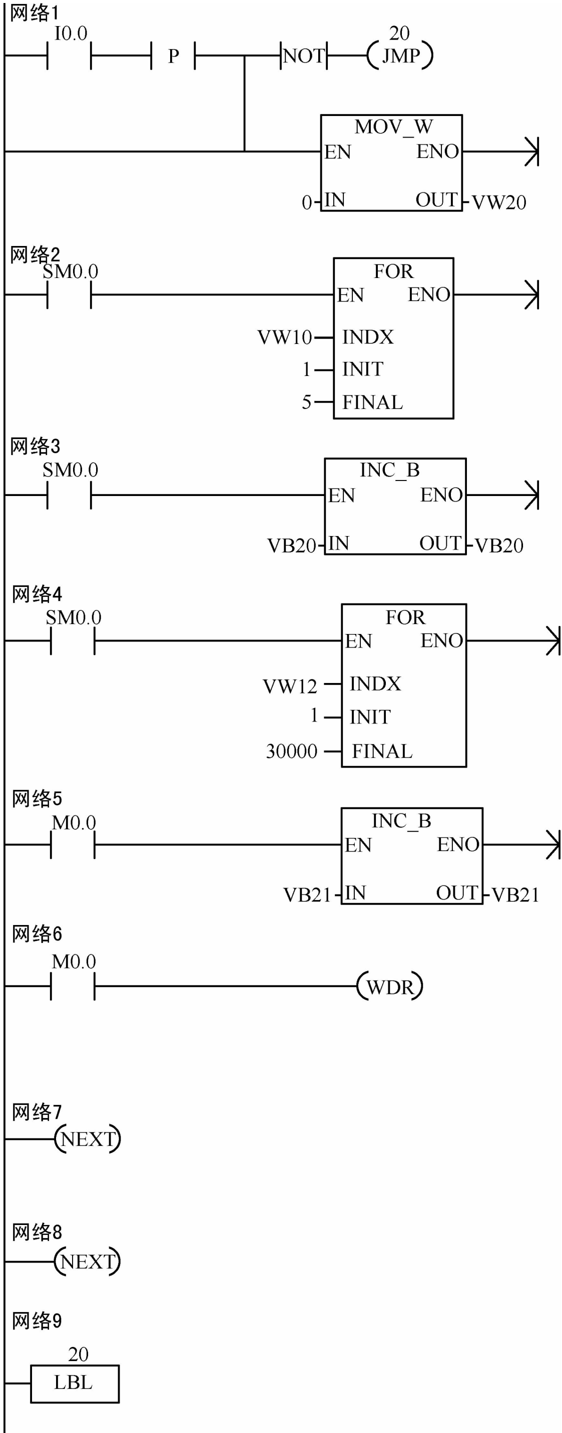


图 12-13　WDR 指令应用

而使用 WDR 指令，可能使控制不准或者失控，导致发生更加严重的事故。

PLC信息

操作模式： 运行

版本
PLC： CPU 226 REL 01.23
固件： 01.23
ASIC: 01.00

扫描速率（ms）
上次： 1
最小值： 1
最大： 14883

错误
严重： 0 不存在严重错误。
非严重： 0 不存在非严重错误。
最后一个严重错误： 3 扫描监视器超时错误。
总计严重错误： 3

I/O错误
错误数目： 1
报告的错误： 存在I/O错误

模块	类型	入	开始	出	开始	状态
PLC	离散	24	I0.0	16	Q0.0	无错误
0	CP243-1 INTERNET	0	I0.0	8	Q2.0	CP243-1的配置失败。
1	EM277 ProfibusDP					无错误
2						不存在
3						不存在
4						不存在
5						不存在
6						不存在

EM信息...　重设扫描速率　关闭

事件历史...

图 12-14　CPU 信息

第 13 章　子程序中断程序库指令

13.1　CALL/RET

问：S7-200 CPU 中有几种程序？ 这些程序什么时候运行？ S7-200 CPU 中有多少子程序？ 怎样使用子程序？ 使用子程序需要注意些什么？

问 1：S7-200 CPU 中有几种程序？

答：在 S7-200 CPU 中，有 4 种程序：OS 系统程序、OB1 主程序、SBR 子程序和 INT 中断程序。OB1 主程序、SBR 子程序和 INT 中断程序也称为用户程序。

问 2：这些程序什么时候运行？

答：这些程序运行规律描述如下：

当 CPU 工作在 STOP 模式，只执行 OS 系统程序。当 CPU 工作在 RUN 模式时，既循环执行 OS 系统程序，又循环执行 OB1 主程序。

在执行程序时当满足调用 SBR 子程序时，会跳至执行 SBR 子程序，执行完 SBR 子程序后，再返回原来的地方继续执行原来的程序。可以从 OB1 主程序、另一个子程序或中断程序调用子程序；但不能从子程序本身调用子程序。在主程序中可以嵌套子程序（在子程序中调用另一个子程序），最大嵌套深度为 8。

当已经声明允许中断，而且有中断事件产生，会暂停执行现在的用户程序（包含 OB1 主程序和 SBR 子程序），进入声明与该中断事件关联的 INT 中断程序，中断程序执行完毕自动返回到原来的地方继续执行原来的程序。

问 3：S7-200 CPU 中有多少子程序？

答：S7-200 CPU 中总共有 64 个子程序（0～63）【CPU 226XM 有 128 个子程序（0～127）】。

问 4：怎样使用子程序？

答：使用子程序可以直接调用，也可以带参数调用。分别描述如下：

在图 13-1～图 13-3 所示的程序中，I0.2 与 Q0.0 的关系是，每当 I0.2 接通一次对 Q0.0 取反控制一次；I0.3 与 Q0.1 的关系是，每当 I0.3 接通一次对 Q0.1 取反控制一次。

当 I0.2 接通瞬间会调用子程序 0，在 I0.0 接通瞬间执行用户的程序流程是：主程序网络 1→子程序 0 网络 1→主程序网络 2→主程序网络 3。

当 I0.3 接通瞬间时会调用子程序 1，当 Q0.1 =0 的条件下在 I0.3 接通瞬间执行用户的程序流程是：主程序网络 1→主程序网络 2→子程序 1 网络 1→子程序 1 网络 2→主程序网络 3。

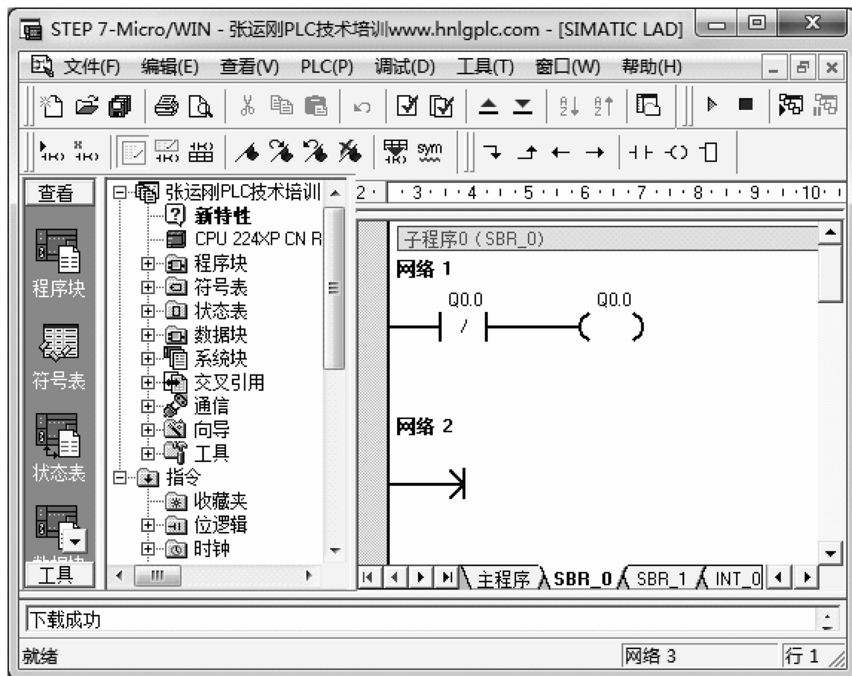


图 13-1　SBR _ 0 子程序 0

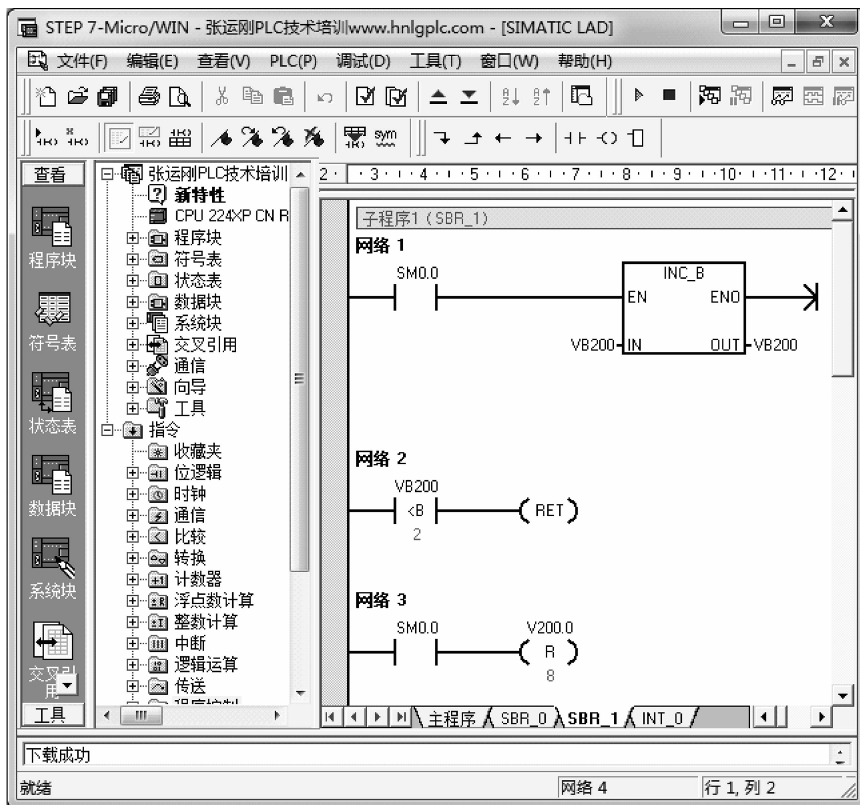


图 13-2　SBR _ 1 子程序 1

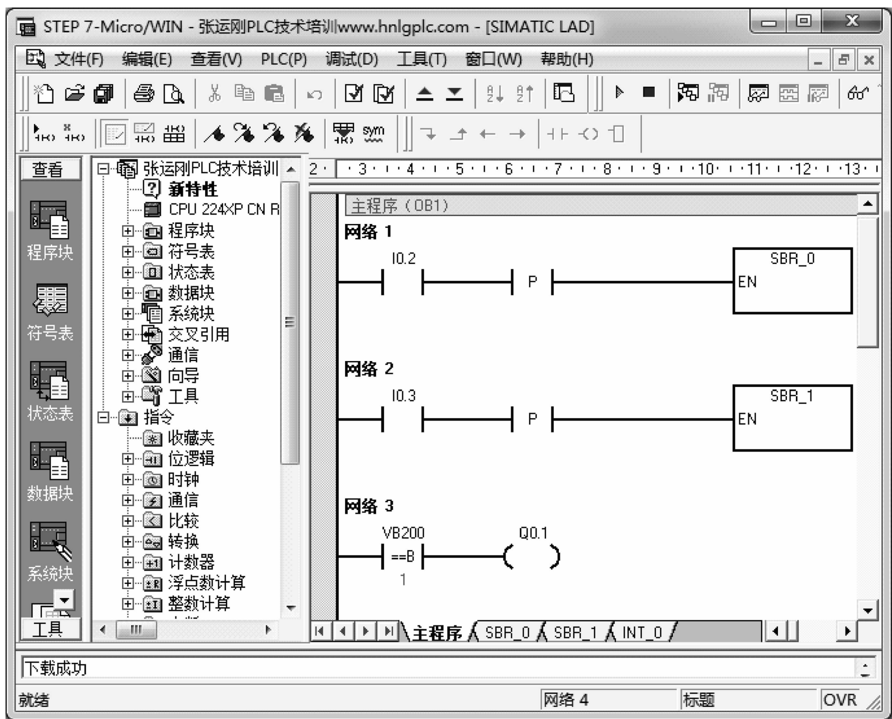


图 13-3　主程序

当 I0.3 接通瞬间时会调用子程序 1，当 Q0.1 =1 的条件下在 I0.3 接通瞬间执行用户的程序流程是：主程序网络 1→主程序网络 2→子程序 1 网络 1→子程序 1 网络 2→子程序 1 网络 3→主程序网络 3。

子程序嵌套使用情况描述如下。

在图 13-4 ~ 图 13-6 所示的程序中，当 I0.0 和 I0.1 都没有接通时，执行用户的程序流程是：主程序网络 1→主程序网络 2→主程序网络 3→主程序网络 1。

当 I0.0 上升沿和 I0.1 =0 时，在 I0.0 接通上升沿扫描周期执行用户的程序流程是：主程序网络 1→子程序 0 网络 1→子程序 0 网络 2→子程序 0 网络 3→主程序网络 2→主程序网络 3。

当 I0.0 上升沿、I0.1 =1 和 Q0.1 =0 时，在 I0.0 接通上升沿扫描周期执行用户的程序流程是：主程序网络 1→子程序 0 网络 1→子程序 0 网络 2→子程序 1 网络 1→子程序 1 网络 2→子程序 0 网络 3→主程序网络 2→主程序网络 3。

当 I0.0 上升沿、I0.1 =1 和 Q0.1 =1 时，在 I0.0 接通上升沿扫描周期执行用户的程序流程是：主程序网络 1→子程序 0 网络 1→子程序 0 网络 2→子程序 1 网络 1→子程序 1 网络 2→子程序 1 网络 3→子程序 0 网络 3→主程序网络 2→主程序网络 3。

带参数调用子程序应用介绍。

在调用子程序时可以带参数调用。参数在子程序的局部变量表中定义。每一个参数需要定义一个符号名（最多为 23 个字符），选择变量类型和数据类型。在带参数调用子程序时可向子程序交接 16 个参数或从子程序交接 16 个参数。

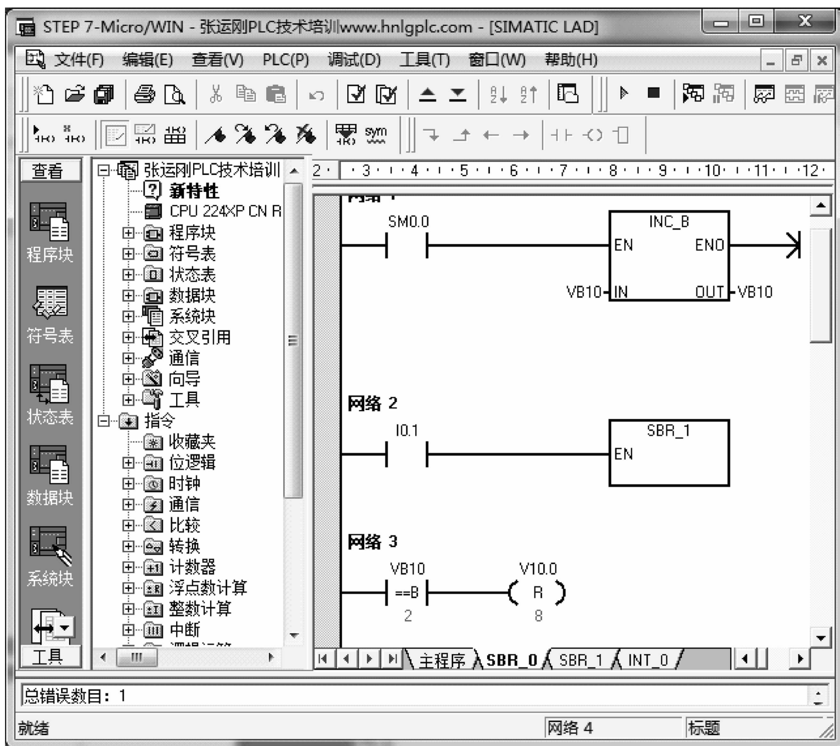


图 13-4　SBR _0 子程序 0

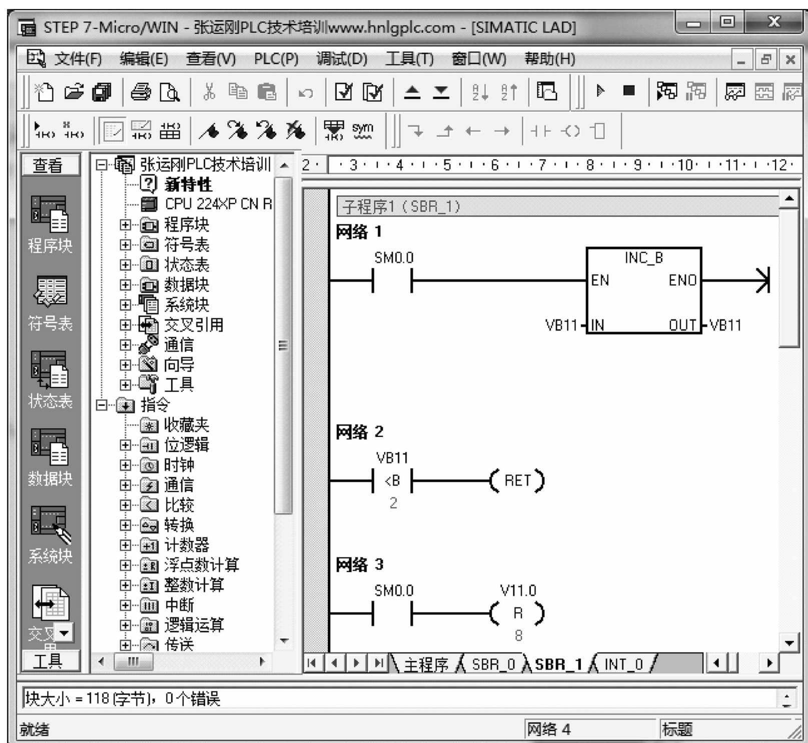


图 13-5　SBR _1 子程序 1

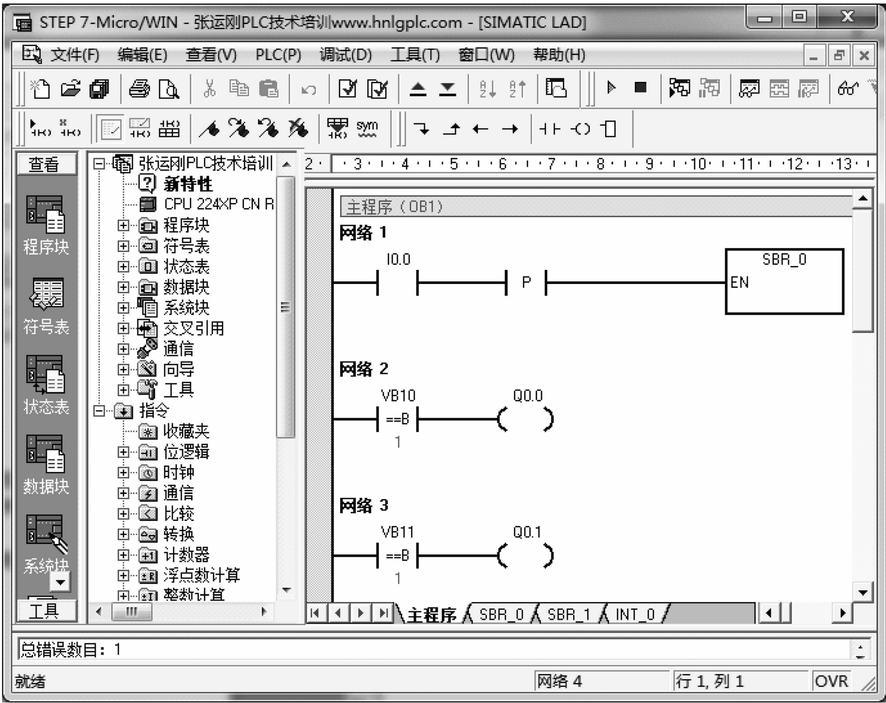


图 13-6　主程序

子程序的局部变量表中的变量类型区定义交接参数和临时参数：IN、IN _ OUT、OUT 和 TEMP。表 13-1 是子程序的参数类型说明。

表 13-1　子程序的参数类型

参数类型		说　明
交接参数	IN	被交接至子程序的参数 可以是直接地址（如 VB1）、间接地址（如 * VD2）、常量（如 12）、地址（如 &VB10）
	IN _ OUT	既是被交接至子程序的参数，又是从子程序交接出来的参数 可以是直接地址（如 VB1），也可以是间接地址（如 * VD2）
	OUT	从子程序的参数交接出的参数 可以是直接地址（如 VB1），也可以是间接地址（如 * VD2）
临时参数	TEMP	用于在子程序中的临时存储

子程序局部变量表中参数的数据类型见表 13-2。

表 13-2　子程序局部变量表中参数的数据类型

参数数据类型	说　明
布尔（BOOL）	布尔数据类型用于位的输入和输出
字节、字、双字（BYTE、WORD、DWORD）	字节、字、双字分别是 8 位、16 位和 32 位不带符号的输入或输出参数
整数、双整数（INT、DINT）	整数、双整数分别是 16 位和 32 位带符号的输入或输出参数

（续）

参数数据类型	说　　明
实数（REAL）	实数数据类型识别单精度（32 位）IEEE 浮点数值
字符串（STRING）	字符串数据类型被用作 32 位指针

在图 13-7 和图 13-8 所示的程序中，L0.0（启动）、L0.1（停止）是被交接至子程序的参数（IN），数据类型是布尔。L0.2（电动机）既是被交接至子程序的参数，又是从子程序交接出来的参数（IN _ OUT），数据类型是布尔。

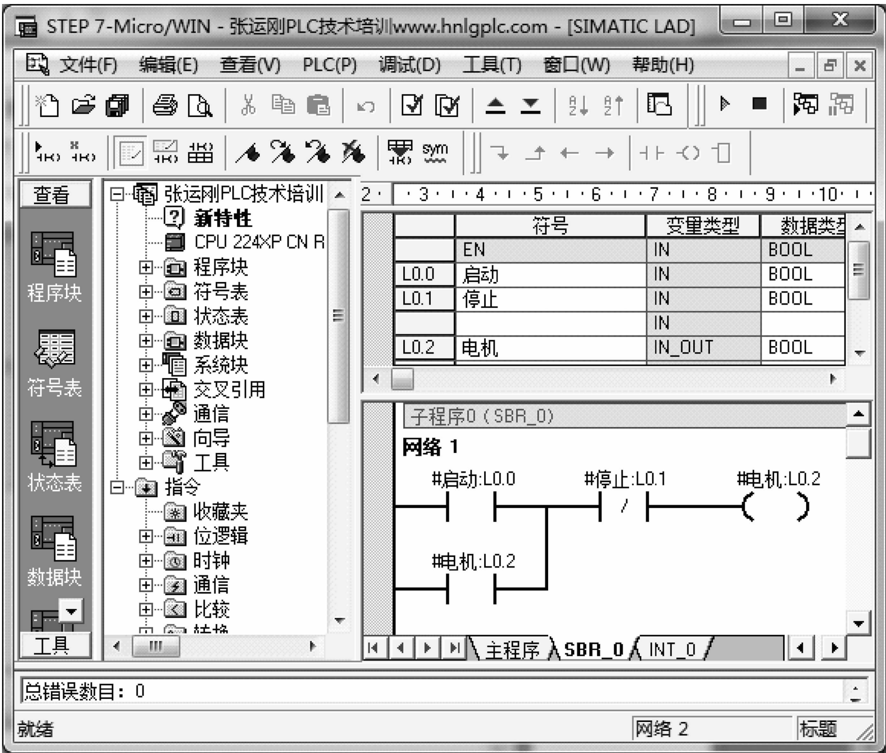


图 13-7　SBR _ 0 子程序 0

问 5：使用子程序需要注意些什么？

答：在使用子程序时需要注意以下几项：

一是在子程序中避免使用 END 指令和 RETI，否则会出现下载编译错误。

二是在子程序嵌套使用情况时，避免嵌套超过 8 层。

三是在子程序中一些计数器和定时器将不能正常工作。

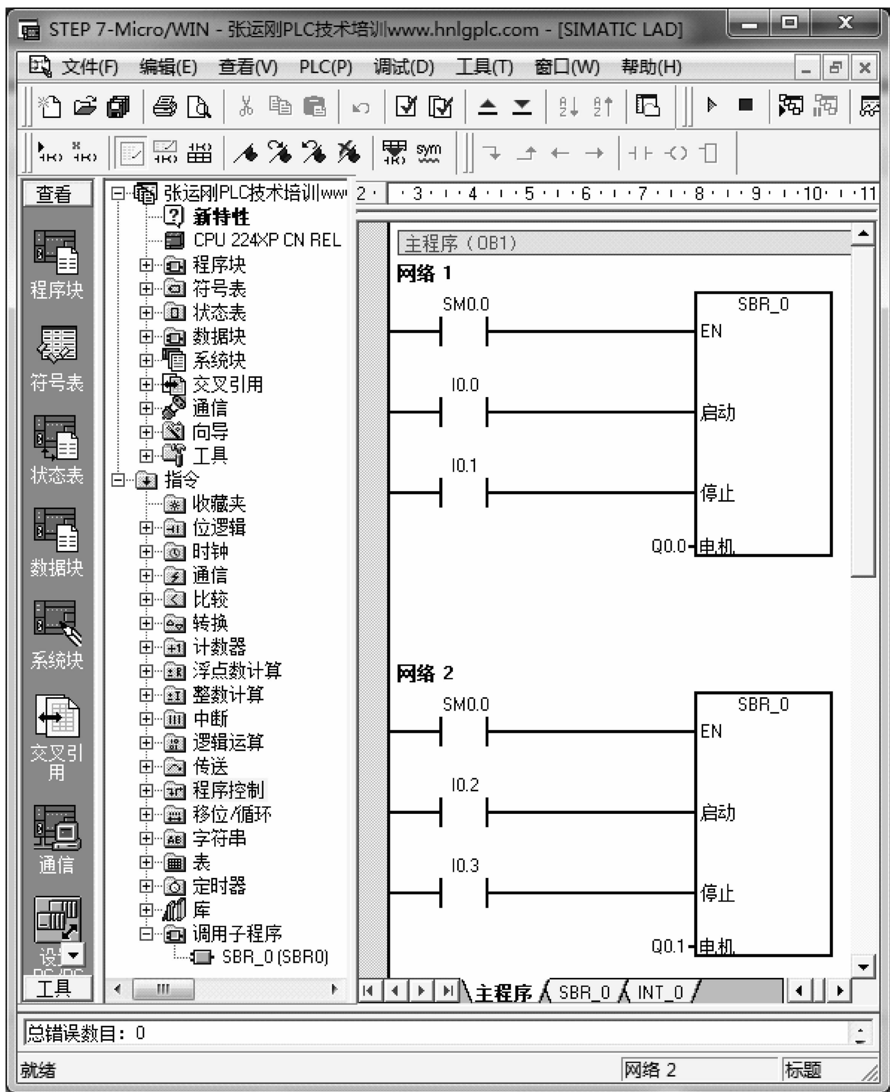


图 13-8　主程序

13.2　中断程序

> 问：S7-200 CPU 中的中断程序有几种？
> 中断指令有哪些？
> 怎样使用中断程序？
> 使用中断程序时需要注意些什么？

问 1：S7-200 CPU 中的中断程序有几种？

答：中断程序按照中断源不同分，有硬件中断、时间中断、高速计数器中断、发完脉冲中断和通信中断等 6 种。S7-200 的中断源使用中断事件来表示，中断事件见表 13-3。

表 13-3　CPU22x 的中断事件表

事件号	说明	优先组	组内优先级别	221、222	224	224XP、226、226XP
0	I0.0 上升沿	中	2	√	√	√
1	I0.0 下降沿		6	√	√	√
2	I0.1 上升沿		3	√	√	√
3	I0.1 下降沿		7	√	√	√
4	I0.2 上升沿		4	√	√	√
5	I0.2 下降沿		8	√	√	√
6	I0.3 上升沿		5	√	√	√
7	I0.3 下降沿		9	√	√	√
8	PORT0 接收字符	高	0	√	√	√
9	PORT0 发送字符		0	√	√	√
10	定时中断 SMB34	低	0	√	√	√
11	定时中断 SMB35		1	√	√	√
12	HSC0 CV = PV	中	10	√	√	√
13	HSC1 CV = PV		13	—	√	√
14	HSC1 计数方向改变		14	—	√	√
15	HSC1 外部复位		15	—	√	√
16	HSC2 CV = PV		16	—	√	√
17	HSC2 计数方向改变		17	—	√	√
18	HSC2 外部复位		18	—	√	√
19	PLS0 脉冲发完		0	√	√	√
20	PLS1 脉冲发完		1	√	√	√
21	定时器 T32 CT = PT	低	2	√	√	√
22	定时器 T96 CT = PT		3	√	√	√
23	PORT0 接收完成	高	0	√	√	√
24	PORT1 接收完成		1	√	√	√
25	PORT1 接收字符		1	—	—	√
26	PORT1 发送字符		1	—	—	√
27	HSC0 计数方向改变	中	11	√	√	√
28	HSC0 外部复位		12	√	√	√
29	HSC4 CV = PV		20	√	√	√
30	HSC4 计数方向改变		21	√	√	√
31	HSC4 外部复位		22	√	√	√
32	HSC3 计数方向改变		19	√	√	√
33	HSC3 外部复位		23	√	√	√

问 2：中断指令有哪些？

答：中断指令见表 13-4。

表 13-4　中断指令表

ATCH	DTCH	ENI	DISI	RETI	CLR _ EVNT
中断连接	中断分离	允许中断	禁止中断	中断返回	清除中断事件

ATCH 和 DTCH 指令样式如图 13-9 ~ 图 13-11 所示。

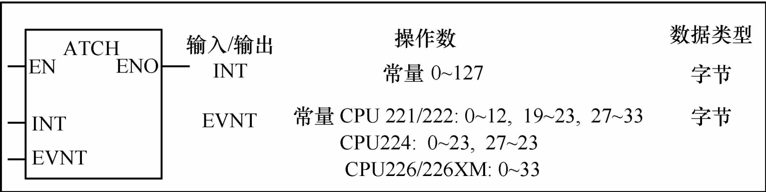


图 13-9　ATCH 指令样式

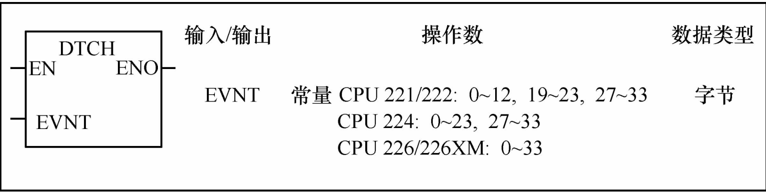


图 13-10　DTCH 指令样式

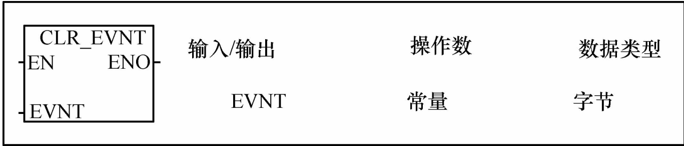


图 13-11　CLR _ EVNT 指令样式

问 3：怎样使用中断程序？

答：中断由事件驱动。在应用中断程序之前，必须使用 ATCH 指令把中断事件与发生该事件的中断程序建立联系，然后使用 ENI 允许开中断。当这个中断事件发生时，会自动执行与该中断事件关联的中断程序。

如果执行了全局禁止中断指令 DISI，将禁止所有的中断，中断事件的每次出现均被排队等候，直至使用全局开中断指令 ENI 重新允许开中断。

如果需要暂停某个中断程序的执行而不影响其他中断程序，可以使用 DTCH 指令声明分离中断事件与中断程序之间的联系。DTCH 指令使中断返回未激活或被忽略状态（被分离的中断事件即使发生，也不会入队等待处理，而是被忽略）。当需要再次声明建立联系时，可以使用 ATCH 指令把中断事件与发生该事件的中断程序建立关联。

使用 ATCH 指令把中断事件与发生该事件的中断程序建立联系，并且声明允许开中断，

当中断事件发生时，立刻执行中断程序（假设其他中断没有执行），一直执行到中断程序的最后一条指令（无条件返回），返回至主程序（或子程序）。如果需要中途返回，可以使用执行“从中断有条件返回”指令（RETI）的方法提前退出中断程序。

由于中断程序是为特殊内部或外部事件提供快速反应而设置的，在使用中断程序时应当尽可能优化中断例行程序，快速执行某项具体任务，然后将控制返回至主程序。通过将中断程序编写得简明扼要，可加快执行的速度，使其他程序不会受到长时间的延误。如果未能做到这一点，无法预料的情形可能导致主程序控制装置出现非正常操作状况。

一个程序中总共可有 128 个中断程序。在各自的优先赋值范围内，PLC 采用先来先服务的原则为中断提供服务。在任何时刻，只能执行一个用户中断程序。一旦一个中断程序开始执行，则一直执行至完成，不会被另一个中断程序所打断，即使是更高优先级别的中断程序。正在处理另一个中断时发生的中断会入队等待处理，每个中断队列可能的最大数目见表 13-5。

表 13-5　每个中断队列最大条目数

	221、222、224	224XP、226、226XP
通信中断队列	4	8
I/O 中断队列	16	16
定时中断对列	8	8

当出现的中断数目超出队列能够容纳的数目时，队列溢出内存位会改变状态，中断队列溢出位见表 13-6。只能在中断程序中使用这些位，因为当队列排空或控制返回主程序时这些位会被复位，所以请在中断程序中使用状态位 SM4. 0、SM4. 1 和 SM4. 2。

表 13-6　中断队列溢出标志位

中断类型	SM 地址	说　　明
通信中断队列	SM4. 0	通信中断队列溢出时该位 =1
I/O 中断队列	SM4. 1	输入中断队列溢出时该位 =1
定时中断对列	SM4. 2	定时中断队列溢出时该位 =1

下面以一个硬件中断，描述中断指令的基本动作和中断的编程方法。

程序如图 13-12 和图 13-13 所示。程序调试操作详细描述如下：

声明中断连接。在图 13-12 所示的程序中，在 CPU“RUN”模式下，接通一次 I0. 4 声明中断程序 0 与中断事件 5 关联，中断事件 5 代表 I0. 2 下降沿产生的中断。

允许开中断。当 I0. 4 接通后，在使 I0. 5 产生下降沿允许开中断，这时监控到 SM4. 4 = 1，说明程序允许响应中断事件。

响应中断执行中断程序。声明中断连接并允许开中断后，让 I0. 2 产生下降沿一次，监控到 VB100 和 VB101 的数据均可加一，说明响应了中断并执行了中断程序 0。

中断提前返回。声明中断连接并允许开中断后，让 I0. 3 =1，这时让 I0. 2 产生下降沿一

次，监控 VB100 和 VB101 的数据变化情况，看到 VB100 加一而 VB101 没有加一，说明执行中断程序 0 时，提前结束并返回。

禁止中断和中断事件排队。声明中断连接并允许开中断后，当尝试接通 I0.6 后，再使 I0.2 产生下降沿一次，监控到 VB100 和 VB101 的状态不变，监控 SM4.4 的状态变为“0”，说明现在是禁止全局中断的状态。这时尝试让 I0.5 产生下降沿，监控到 SM4.4 的状态变为“1”，监控到 VB100 的数据会增加，说明在禁止全局中断的时候，I0.2 的下降沿中断进入列队等候处理，当允许开全局中断时，立刻进入中断程序执行任务。

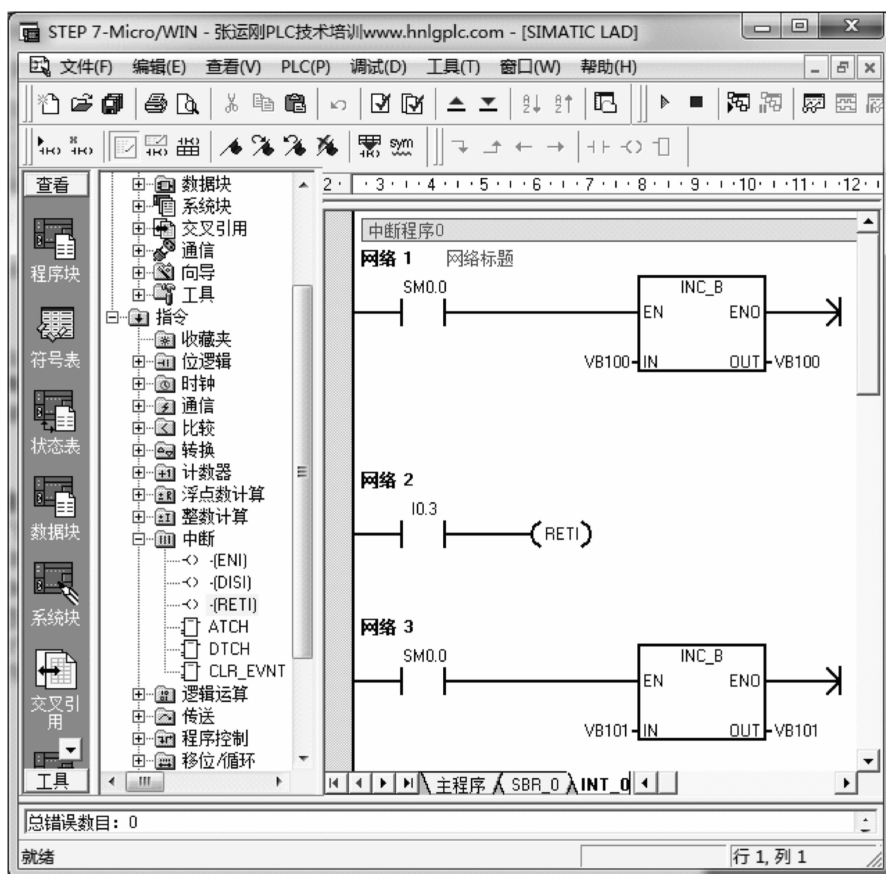


图 13-12　INT _0 中断程序 0

清除中断事件。声明中断连接并允许开中断后，当尝试接通 I0.6 后，再使 I0.2 产生下降沿一次，监控到 VB100 和 VB101 的状态不变，监控 SM4.4 的状态变为“0”，说明现在是禁止全局中断的状态，这时 I0.2 下降沿产生的中断事件会在排队。这时接通 I1.0 把排队的中断事件清零后，这时尝试让 I0.5 产生下降沿，监控到 SM4.4 的状态变为“1”，监控到 VB100 的数据没有变化，说明在排队的中断事件被清零了。

中断分离。声明中断连接并允许开中断后，当尝试接通 I0.6 后，再使 I0.2 产生下降沿

一次，监控到 VB100 和 VB101 的状态不变，说明中断已经分离。如果需要重新声明连接可以接通 I0.4 一次。

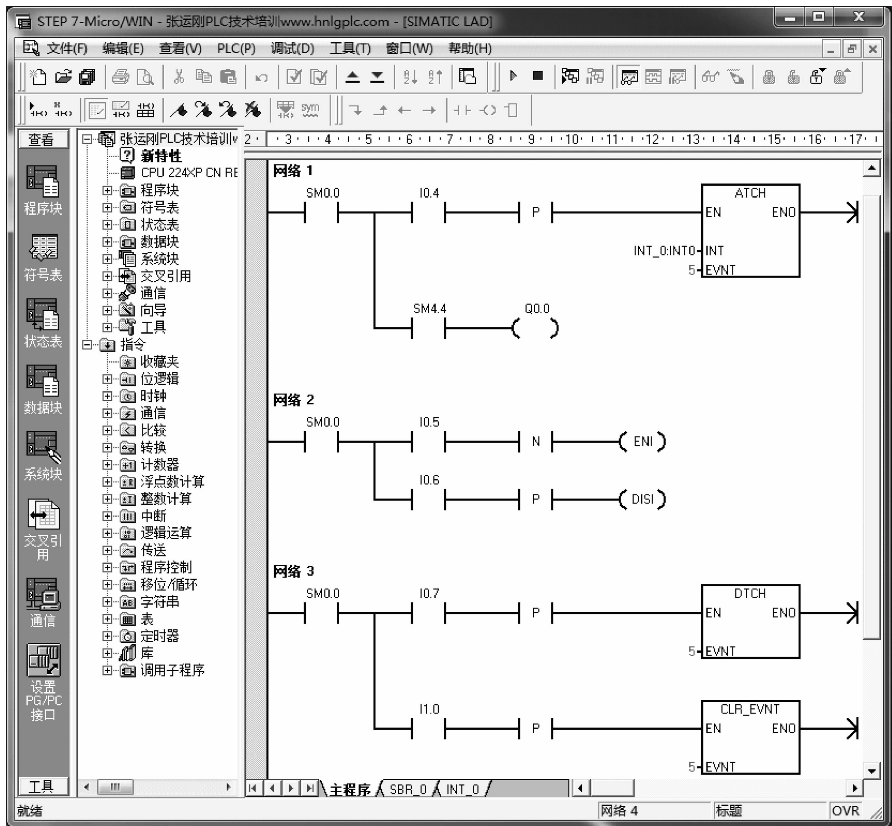


图 13-13　主程序

时间中断程序应用例如图 13-14 和图 13-15 所示。当运行程序时，可以看到 Q0.0 接通 1s，断开 1s，这样的规律一直循环往复。

高速计数器中断、脉冲输出中断和通信中断请查看本书其他相关章节的内容。

问 4：使用中断程序时需要注意些什么？

答：在使用中断程序时需要注意以下几项：

一是在中断程序中不能使用 DISI、ENI、HDEF、SCR、END 和 RET 指令。

二是 SM4.0、SM4.1 和 SM4.2 这些标志位请在中断程序中使用。

三是注意中断事件队列溢出。

四是中断程序的累加器与主程序和子程序的是完全独立的两套累加器。

五是计数器和一些定时器在中断程序中无法正确执行。

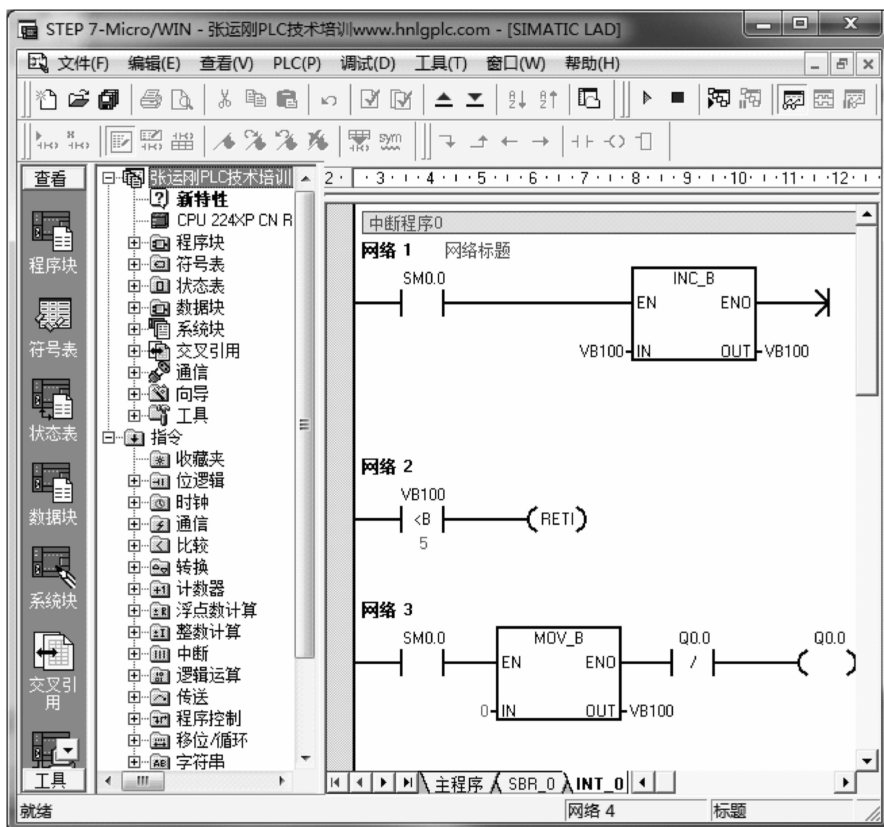


图 13-14　中断程序 0

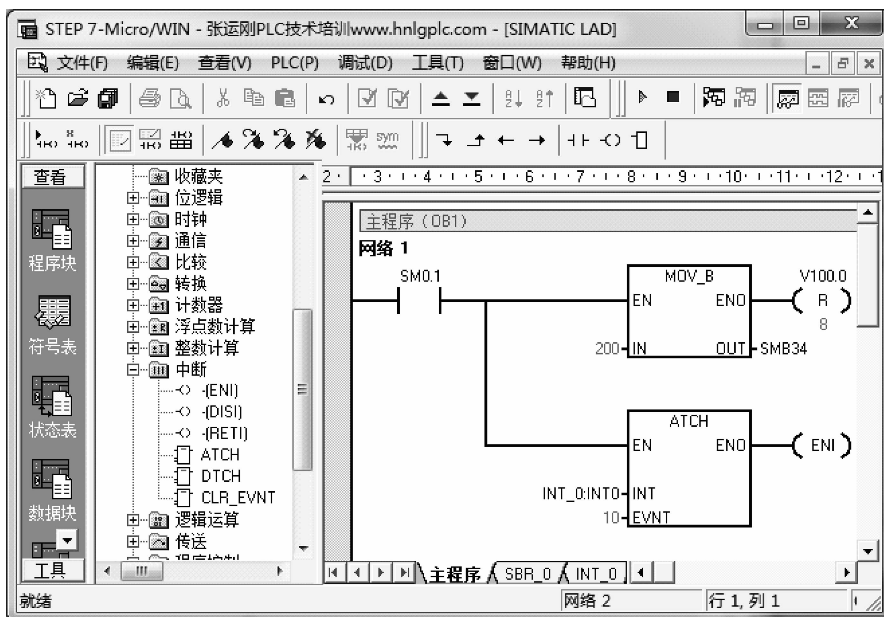


图 13-15　主程序

第 14 章　高速计数器

> 问：高速计数器指令有哪些？
> S7-200 支持几个高速计数器？
> S7-200 高速计数器各种计数模式的输入 I 分配情况如何？
> 这些高速计数器控制字分配情况怎样？
> 这些高速计数器定义初值和目标值情况怎样？
> 监控这些高速计数器的状态字分配情况怎样？
> 如何理解高速计数器和应用高速计数器？

问 1：高速计数器指令有哪些？

答：高速计数器指令有：HDEF 定义高速计数器指令和 HSC 激活高速计数器指令，这些指令样式如图 14-1 和图 14-2 所示。

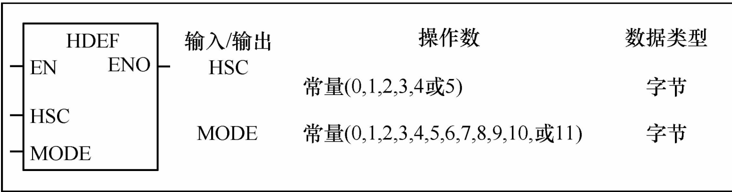


图 14-1　HDEF 指令样式

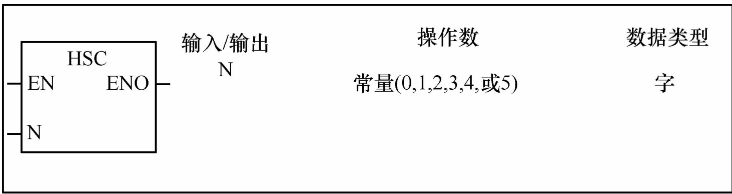


图 14-2　HSC 指令样式

问 2：S7-200 支持几个高速计数器？

答：CPU 型号不同，支持的高速计数器个数不同，表 14-1 列举了各种 CPU 支持的高速计数器情况，“✓”表示支持，“×”表示不支持。

表 14-1　各种 CPU 支持的高速计数器

	HSC5	HSC4	HSC3	HSC2	HSC1	HSC0
CPU221 CPU222	✓	✓	✓	×	×	✓
CPU224 CPU224XP CPU226	✓	✓	✓	✓	✓	✓

问 3：S7-200 高速计数器各种计数模式的输入 I 分配情况如何？

答：每个高速计数器均已配置固定的输入端作为高速计数器的脉冲输入、计数方向、启动和复位端子。同一个计数器选择的计数模式不同，系统分配的固定输入端的数目和功能可能不同，详细情况见表 14-2。

表 14-2　各个高速计数器输入端的分配表

			输入端的分配			
		HSC0	I0.0	I0.1	I0.2	
		HSC1	I0.6	I0.7	I1.0	I1.1
		HSC2	I1.2	I1.3	I1.4	I1.5
		HSC3	I0.1			
模式	类型	HSC4	I0.3	I0.4	I0.5	
		HSC5	I0.4			
0	单相计数器	内部方向控制	计数脉冲			
1			计数脉冲		复位	
2			计数脉冲		复位	启动
3		外部方向控制	计数脉冲	方向控制		
4			计数脉冲	方向控制	复位	
5			计数脉冲	方向控制	复位	启动
6	双相计数器	两计数	加计数脉冲	减计数脉冲		
7			加计数脉冲	减计数脉冲	复位	
8			加计数脉冲	减计数脉冲	复位	启动
9		A/B 相计数	A 计数脉冲	B 计数脉冲		
10			A 计数脉冲	B 计数脉冲	复位	
11			A 计数脉冲	B 计数脉冲	复位	启动
12	内部计数	HSC0 计数 Q0.0 的输出脉冲，HSC3 计数 Q0.1 的输出脉冲				

问 4：这些高速计数器控制字分配情况怎样？

答：使用高速计数器前，必须要使能相应的控制字，不同高速计数器均已配置固定的控制，这些控制详细情况见表 14-3。

表 14-3　各个控制字说明

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	说　明		
SMB37	SMB47	SMB57	SMB137	SMB147	SMB157	功能	设置数字	默认值
SM37.0	SM47.0	SM57.0	—	SM147.0	—	复位	=0 高电平	=0 高电平
							=1 低电平	
—	SM47.1	SM57.1	—	—	—	启动	=0 高电平	=0 高电平
							=1 低电平	
SM37.2	SM47.2	SM57.2	—	SM147.2	—	正交计数率	0 =4 ×	0 =4 ×
							1 =1 ×	

（续）

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	说明		
SMB37	SMB47	SMB57	SMB137	SMB147	SMB157	功能	设置数字	默认值
SM37.3	SM47.3	SM57.3	SM137.3	SM147.3	SM157.3	计数方向	0 = 减计数	
							1 = 加计数	
SM37.4	SM47.4	SM57.4	SM137.4	SM147.4	SM157.4	更新计数方向	=0 不更新	
							=1 更新	
SM37.5	SM47.5	SM57.5	SM137.5	SM147.5	SM157.5	更新设定值	0 = 不更新	
							1 = 更新	
SM37.6	SM47.6	SM57.6	SM137.6	SM147.6	SM157.6	更新当前值	0 = 不更新	
							1 = 更新	
SM37.7	SM47.7	SM57.7	SM137.7	SM147.7	SM157.7	启用 HSC	0 = 禁用	
							1 = 启用	

问 5：这些高速计数器定义初值和目标值情况怎样？

答：更新设定值和当前值的特殊内存均已固定分配，见表 14-4。

表 14-4　更新设定值和当前值分配表

更新数值	HSC0	HSC1	HSC2	HSC3	HSC4	HSC5
更新当前值	SMD38	SMD48	SMD58	SMD138	SMD148	SMD158
更新设定值	SMD42	SMD52	SMD62	SMD142	SMD152	SMD162

问 6：监控这些高速计数器的状态字分配情况怎样？

答：在使用高速计数器过程中，可以监控高速计数器的当前值，也可以监控当前计数方向以及当前值是否大于或等于预设值。这些数据和内存位均已固定分配好，见表 14-5。

表 14-5　状态内存位

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	说明	
SMB36	SMB46	SMB56	SMB136	SMB146	SMB156	功能	状态数字
SM36.0	SM46.0	SM56.0	SM136.0	SM146.0	SM156.0	未定义	
SM36.1	SM46.1	SM56.1	SM136.1	SM146.1	SM156.1		
SM36.2	SM46.2	SM56.2	SM136.2	SM146.2	SM156.2		
SM36.3	SM46.3	SM56.3	SM136.3	SM146.3	SM156.3		
SM36.4	SM46.4	SM56.4	SM136.4	SM146.4	SM156.4		

（续）

HSC0	HSC1	HSC2	HSC3	HSC4	HSC5	说　明	
SMB36	SMB46	SMB56	SMB136	SMB146	SMB156	功　能	状态数字
SM36. 5	SM46. 5	SM56. 5	SM136. 5	SM146. 5	SM156. 5	当前计数方向状态位	0 = 减计数
							1 = 加计数
SM36. 6	SM46. 6	SM56. 6	SM136. 6	SM146. 6	SM156. 6	当前值等于预设值状态位	0 = 不相等
							1 = 相等
SM36. 7	SM46. 7	SM56. 7	SM136. 7	SM146. 7	SM156. 7	当前值大于预设值状态位	0 = 小于或等于
							1 = 大于
HC0	HC1	HC2	HC3	HC4	HC5	监控当前值	

只有在执行高速计数器中断程序时，状态位才有效。监控高速计数器状态的目的在于启用对正在执行的操作有重大影响的事件的中断程序。

问 7：如何理解高速计数器和应用高速计数器?

答：使用高速计数器之前必须选择计数器模式，使用 HDEF 指令定义高速计数器计数模式，系统按照用户所用模式分配输入点。然后使用 HSC 指令激活即可计数。

定义了计数器和计数器模式后，可以为计数器配置动态参数编程。每个高速计数器均有一个控制字节，通过设置控制字节可以完成如下功能：

①启用或禁用计数器。

②控制方向（仅限模式 0、1 和 2）或初始化所有其他模式的计数方向。

③更新当前值和设定值。

初始化高速计数器参数一般使用首次扫描特殊继电器内存位 SM0. 1，如果不是这样，在进入 RUN（运行）模式后，只能为每个高速计数器执行一次 HDEF 指令。如果为高速计数器多次执行 HDEF 会生成运行时间错误，并不会改变计数器首次执行 HDEF 时计数器的设置方式。

以下实例说明如何配置当前值、设定值、更改方向和使用高速中断，方法是以希望的方式设置 SMB47（对于 HSC1）数值，然后执行 HSC 指令。

下面以 HSC1 为例介绍使用指令向导初始化 HSC1 工作于模式 5 的方法。

下面按照步骤说明如何为带外部方向的单相向上/向下计数器模式 5 初始化 HSC1：

①使用首次扫描 SM0. 1 调用执行初始化操作的子程序。

②在初始化子例行程序中，根据所需的控制操作载入 SMB47。

例如：SMB47 = 16#F8 产生下列结果：

启用计数器；

写入新当前值；

写入新设定值；

设置 HSC 初始方向，向上计数；

将启动和复位输入设为高电平有效。

③执行 HDEF 指令，HSC 输入设为 1，无外部复位或启动的 MODE（模式）输入设为 3，

有外部复位但无启动设为 4，有外部复位和启动设为 5。

④用所需的当前值载入 SMD48。

⑤用所需的设定值载入 SMD52。

⑥为了捕获当前值等于预设值，将 CV = PV 中断事件（事件 13）附加于中断程序中，为中断编程。

⑦为了捕获方向改变，将方向改变中断事件（事件 14）附加于中断程序中，为中断编程。

⑧为了捕获外部复位事件，将外部复位中断事件（事件 15）附加于中断程序中，为中断编程。

⑨执行全局中断允许指令（ENI），允许开中断。

⑩执行 HSC 指令，使 S7-200 为 HSC1 激活编程。

⑪退出子程序。

在编程界面，打开指令向导，如图 14-3 所示，单击“工具”→“指令向导”，会弹出如图 14-4 所示的指令选择画面，选择“HSC”，然后单击“下一步”。

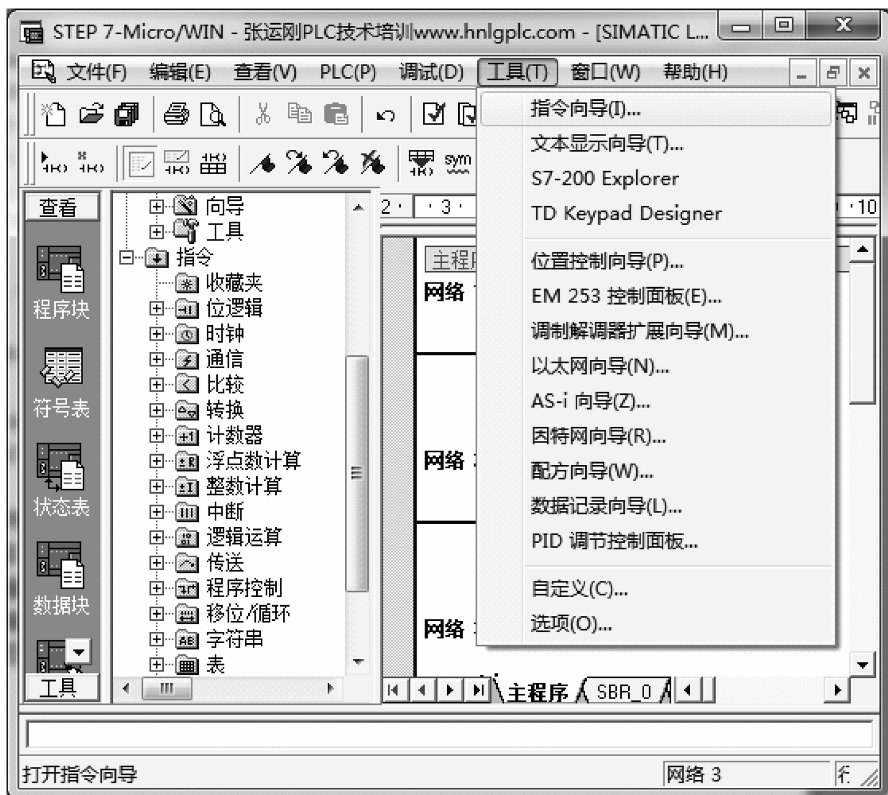


图 14-3　打开指令向导

在图 14-5 中，选择准备使用的 HSC1 和模式 5，从表 14-2 可以看出 HSC1 和模式 5 是单相带外部方向控制的单脉冲高速计数器，同时自带复位端和启动端。计数器脉冲信号输入端系统分配为 I0.6，外部方向控制端系统分配为 I0.7，复位端子是 I1.0，启动端子是 I1.1。

选择完毕后单击“下一步”。

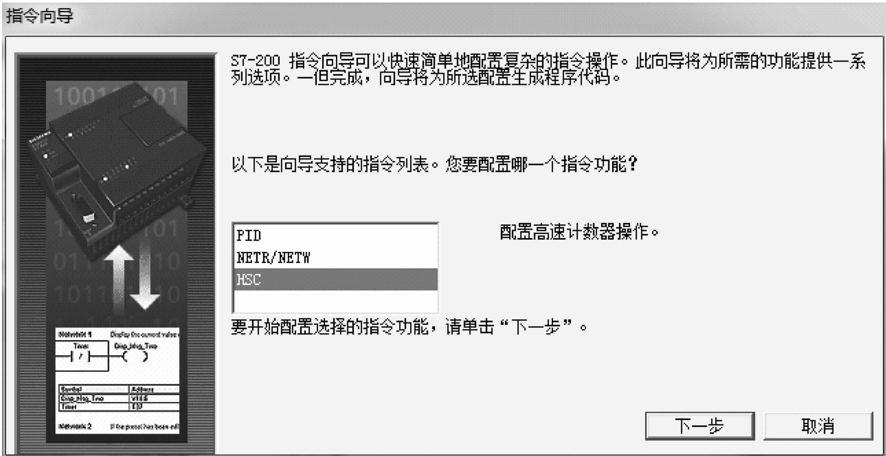


图 14-4　选择功能

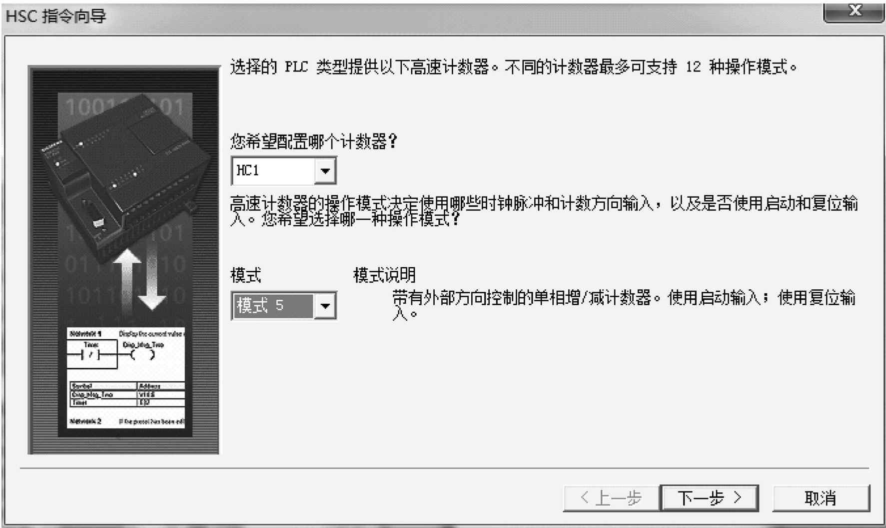


图 14-5　选择高速计数器和模式

在图 14-6 中，可以为 HSC1 初始化用的子程序重新命名或默认程序名称，并写上初始状态，HSC1 的设定值本例是“100”，当前值本例是“0”，计数方向本例选择“向上”（即加计数），复位和启动端子选择高电平有效。选择完毕后单击“下一步”。

在图 14-7 中，声明使用计数方向改变时的中断事件，声明使用外部复位的中断事件，也声明使用计数器经过值达到预设值的中断事件，这些中断事件允许自定义符号本例使用默认的中断程序符号。选择完毕后单击“下一步”。

本例在图 14-7 中，希望为 HC1 编程多少步的选择框里选择“1”，意思是当达到第一次的 CV = PV 时，声明下一步的动态参数。本例为下一步动态参数更改 PV = 50，CV = 0，如图 14-8 所示。

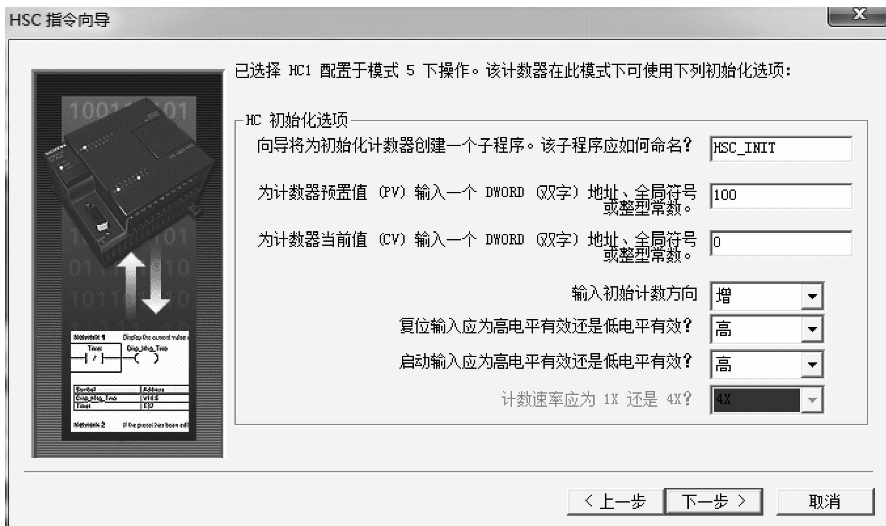


图 14-6　初始化 HSC1

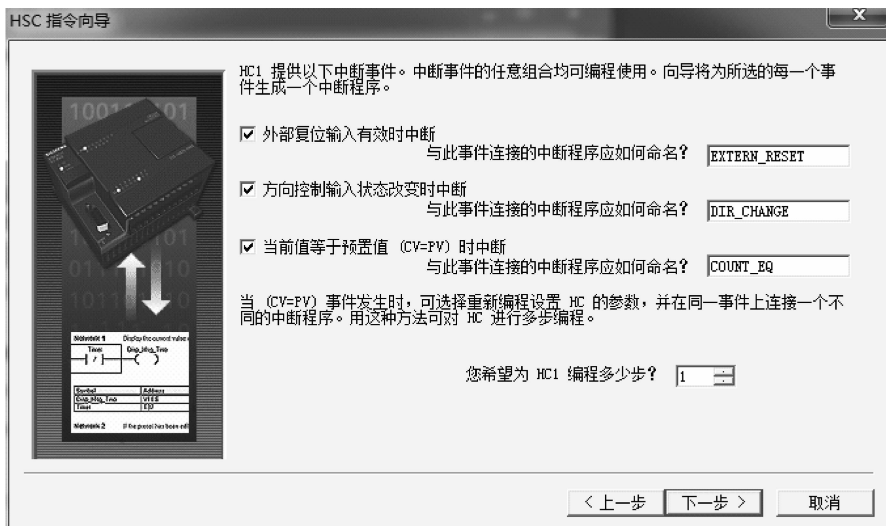


图 14-7　选择中断

在图 14-9 中，可以看到向导按照前面步骤生成的初始化子程序和 3 个中断程序名称。然后单击“完成”，在弹出的确认画面中选择“是”即可。

指令向导生成的初始化子程序 1 梯形图程序如图 14-10 所示。中断事件如图 14-11 ~ 图 14-13 所示。其中图 14-11 ~ 图 14-13 添加了用户为了达到控制任务的用户程序。图 14-14 是主程序，使用开机脉冲完成初始化 HC1 计数器功能。

在图 14-10 ~ 图 14-14 所示指令向导生成的程序和用户编写的程序中，每次启动 CPU，调用初始化子程序 HSC _ INT，HSC1 被配置为模式 5 工作模式，系统自动分配 I0. 6 作为计数脉冲输入端、I0. 7 作为外部计数方向控制端、I1. 0 为复位端和 I1. 1 为启动端，HSC1 的设定值是“100”、当前值是“0”、计数方向选择“向上”即加计数。

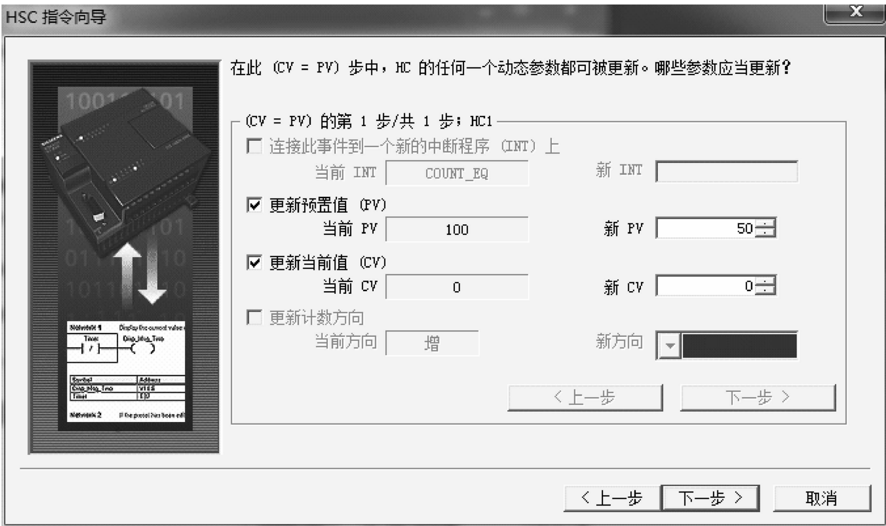


图 14-8　更新动态参数

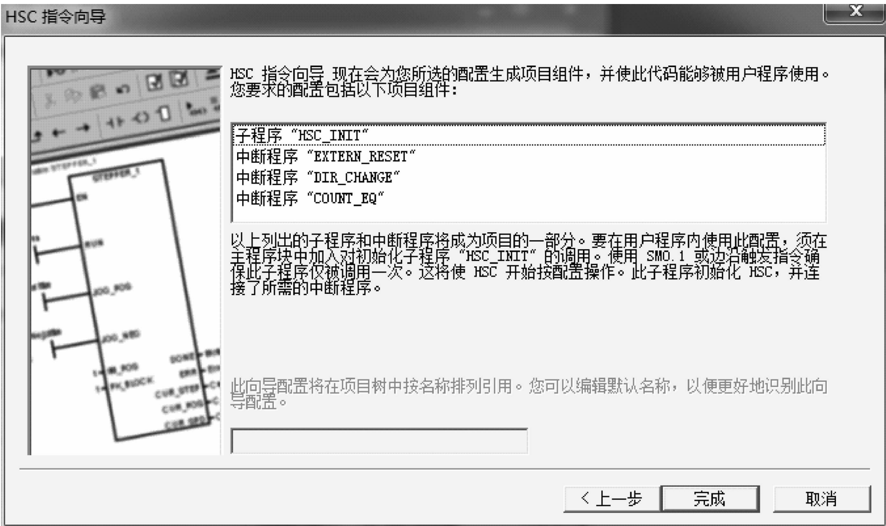


图 14-9　向导生成的组件

调试程序如下：

加计数操作。CPU 工作启动后在 RUN 模式，使 I1.1 = 1 启动计数器，使 I0.7 = 1 控制加计数，然后接通 I0.6 一次，监控 VD600 的数在增加，说明在进行加计数。

减计数操作。CPU 工作启动后在 RUN 模式，使 I1.1 = 1 启动计数器，使 I0.7 = 0 控制减计数，然后接通 I0.6 一次，监控 VD600 的数在减少，说明在进行减计数。

执行 DIR _ CHANGE 中断程序。每当计数器从加计数变为减计数，或者每当计数器从减计数变为加计数时，监控 VB501 的数据都在增加，说明响应了 DIR _ CHANGE 中断程序。

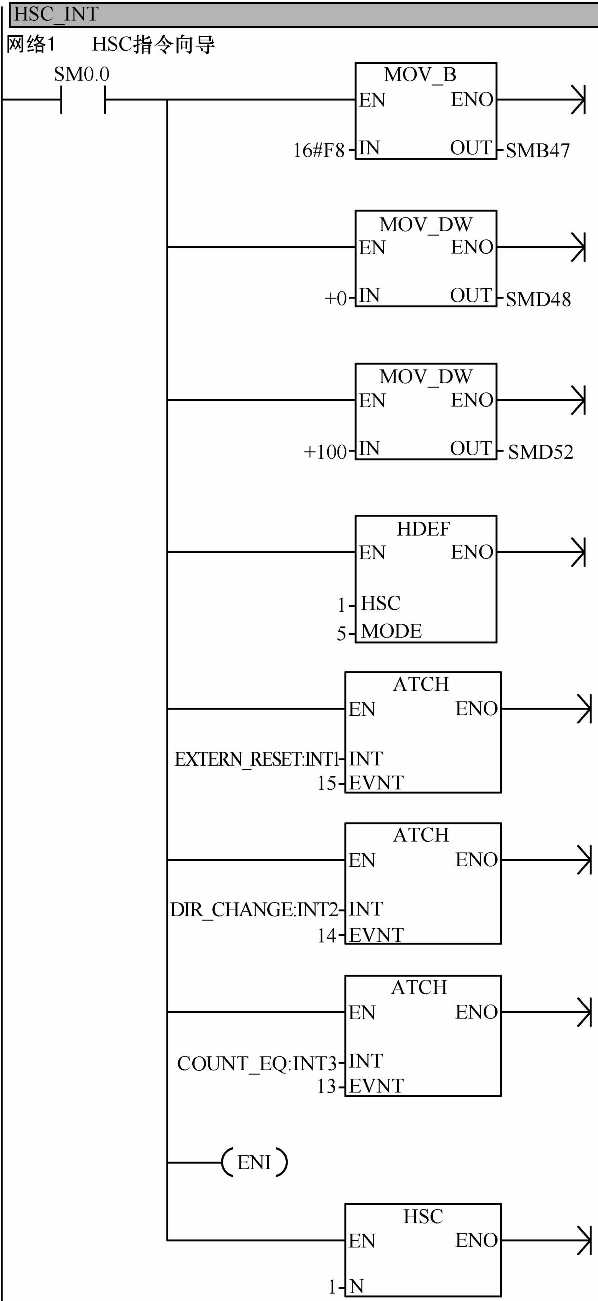


图 14-10 初始化子程序

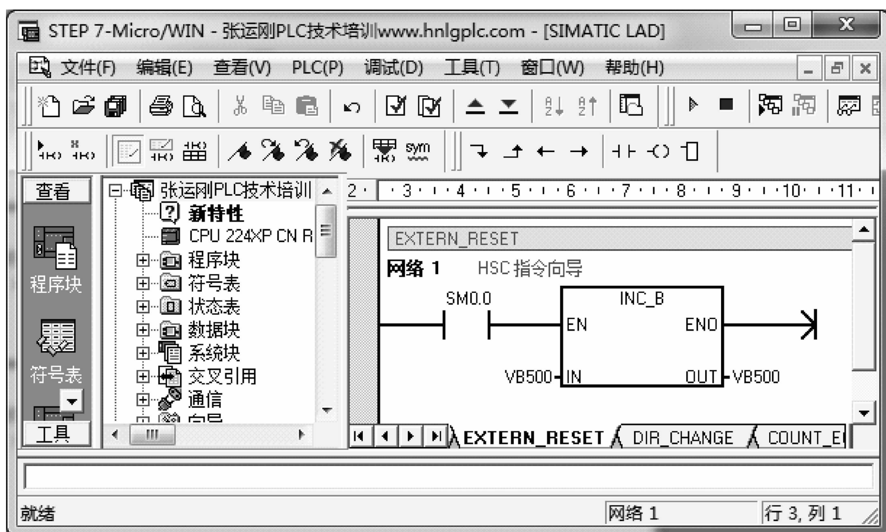


图 14-11　复位中断程序



图 14-12　计数方向改变中断程序

复位操作。CPU 工作启动后在 RUN 模式，使 I1.1 = 1 启动计数器，然后接通 I1.0 一次，监控 VB500 的数据会增加，VD600 的数会变为 0，说明计数器在复位，复位过程中也响应了复位中断程序。

执行 COUNT _ EQ 中断程序。当第一次 CV 值达到 100 时，会执行 COUNT _ EQ 中断程序，程序状态将监控到 VB502 的数值增加了，同时 HC1 的数值变为 0。以后每当 CV 值达到 50 时，会执行 COUNT _ EQ 中断程序。

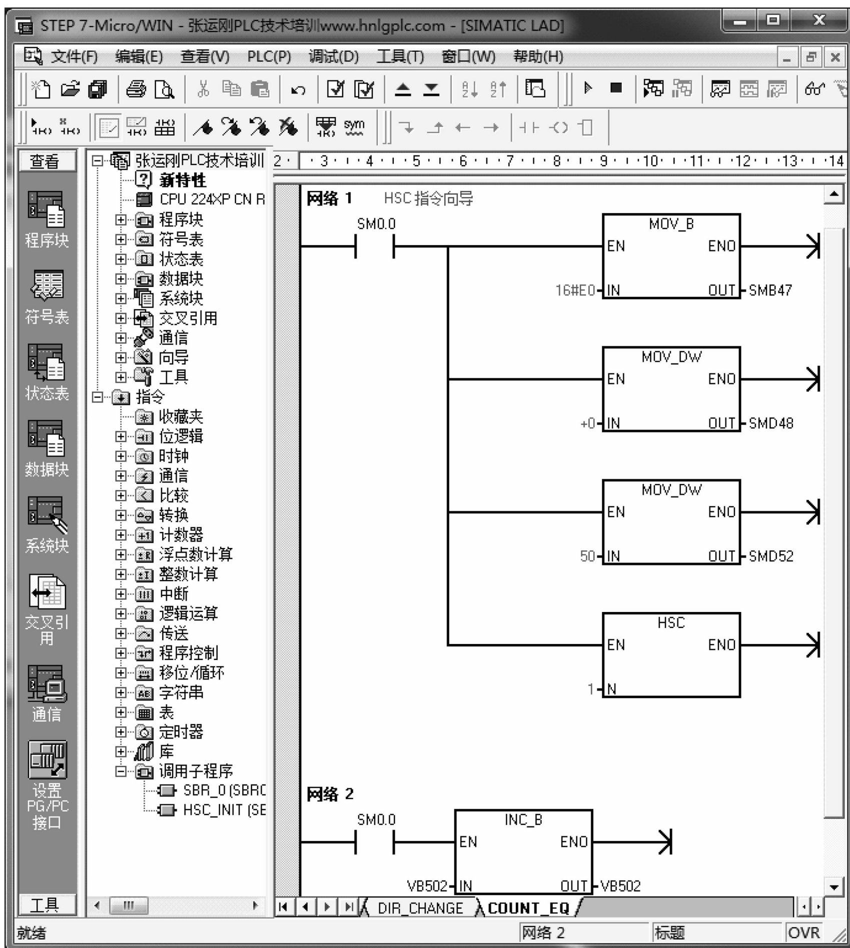


图 14-13　CV = PV 中断程序

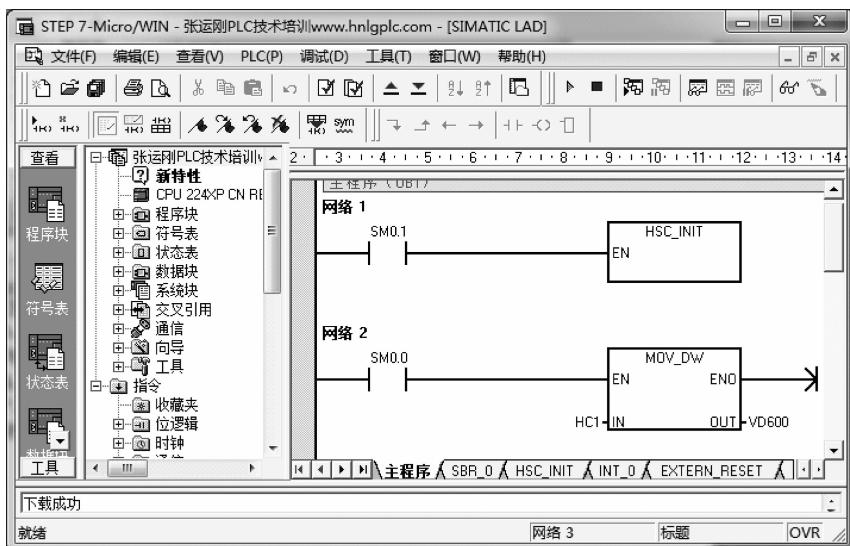


图 14-14　主程序

第 15 章　脉冲输出指令

问：S7-200 CPU 可以发几路脉冲？ PWM 与 PTO 脉冲有什么特征？ 脉冲输出控制字和状态字是什么？ 如何应用发 PWM 脉冲？ 如何应用发单段 PTO 脉冲？ 如何应用发多段 PTO 脉冲？

问 1：S7-200 CPU 可以发几路脉冲？

答：S7-200CPU 可以发两路脉冲，分别从 Q0.0 或 Q0.1 发脉冲，脉冲形式可以是 PWM 或者 PTO 脉冲。

Q0.0 和 Q0.1 在被 PTO 或 PWM 功能使用时，Q0.0 和 Q0.1 由 PTO/PWM 控制输出，并禁止输出点的其他使用，输出波形不受输出映像寄存器状态、强迫数值、执行立即输出指令的影响。PTO/PWM 输出必须至少有 10% 的额定负载，才能完成从关闭至打开以及从打开至关闭的顺利转换。

Q0.0 和 Q0.1 在没有被 PTO 或 PWM 功能使用时，输出控制转交给输出映像寄存器。输出映像寄存器决定输出波形的初始和最终状态。

问 2：PWM 与 PTO 脉冲有什么特征？

答：PWM 可以控制输出一串占空比可调或脉冲周期可调的脉冲，输出波形如图 15-1 所示，在发脉冲的过程中，用户可以改变脉冲的周期和宽度。脉冲周期和宽度的单位可以是 ms 或者 μs（毫秒或者微秒）。

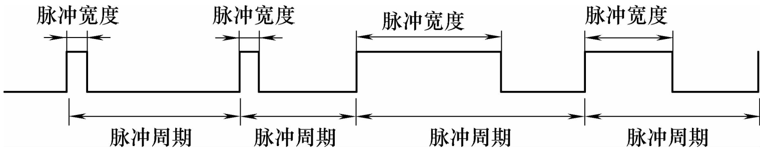


图 15-1　PWM 脉冲

PTO 可以控制输出一串占空比 50% 的脉冲，输出波形如图 15-2 所示，用户可以控制输出预先给定参数的脉冲，这些参数是周期、脉冲个数和周期增量。

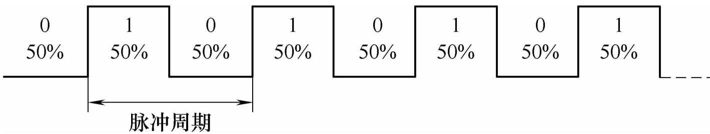


图 15-2　PTO 脉冲

问 3：脉冲输出控制字和状态字是什么？

答：SMB67 控制 Q0.0 发出 PTO 或 PWM 脉冲，SMB77 控制 Q0.1 发出 PTO 或 PWM 脉冲。

通过修改 SM 区（包括控制字节）中的数值，然后执行 PLS 指令，可以改变 PTO 或 PWM 波形的特征。通过向控制字节的 PTO/PWM 启用位（SM67.7 或 SM77.7）写入 0，然后执行 PLS 指令，可以在任何时间禁用 PTO 或 PWM 脉冲的输出。

PTO/ PWM 控制寄存器表描述用于控制 PTO/PWM 操作的寄存器，PTO/PWM 控制寄存器见表 15-1。

表 15-1　PTO/PWM 控制特殊继电器表

Q0.0	Q0.1	状　态　位		
SM66.4	SM76.4	PTO 增量算术运算脉冲周期非法值	0 = 无错	1 = 中止
SM66.5	SM76.5	PTO 由于用户命令而终止	0 = 无终止	1 = 中止
SM66.6	SM76.6	PTO 线管溢出	0 = 无溢出	1 = 溢出
SM66.7	SM76.7	PTO 执行状态	0 = 执行中	1 = 无执行（空闲）
Q0.0	Q0.1	控制字节		
SM67.0	SM77.0	PTO/PWM 更新周期	0 = 禁止更新	1 = 允许更新
SM67.1	SM77.1	PWM 更新脉冲宽度	0 = 禁止更新	1 = 允许更新
SM67.2	SM77.2	PTO 更新脉冲数	0 = 禁止更新	1 = 允许更新
SM67.3	SM77.3	PTO/PWM 时基选择	0 = 1μs	1 = 1ms
SM67.4	SM77.4	PWM 更新方法	0 = 异步更新	1 = 同步更新
SM67.5	SM77.5	PTO 单段/多段选择	0 = 单段操作	1 = 多段操作
SM67.6	SM77.6	PTO/PWM 选择	0 = PTO	1 = PWM
SM67.7	SM77.7	PTO 和 PWM 禁止/允许	0 = 禁止	1 = 允许
Q0.0	Q0.1	其他相关寄存器		
SMW68	SMW78	单段 PTO/PWM 周期时间	（范围：2 ~ 65535）	
SMW70	SMW80	PWM 脉冲宽度	（范围：0 ~ 65535）	
SMD72	SMD82	单段 PTO 脉冲数	（范围：1 ~ 4294967295）	
SMB166	SMB176	多段 PTO 操作中，进行中的段数		
SMW168	SMW178	多段 PTO 操作中，轮廓表的开始位置		

以下 3 种情况会使 SM66.4（或 SM76.4）或 SM66.5（或 SM76.5）特殊继电器位为“1”：

1）指定一个在输出很多脉冲后按照指定的脉冲增量，产生非法脉冲周期值（超出脉冲

周期)，从而产生算术溢出错误，会终止 PTO 功能，并将“增量计算错误”位（SM66.4 或 SM76.4）设为 1。输出控制权交还输出映像寄存器控制。

2）如果以手动方式中止（禁用）正在执行的 PTO 输出，会把“用户中止”位（SM66.5 或 SM76.5）设为 1。

3）尝试在线管已满的情况下再载入线管，PTO 溢出位（SM66.6 或 SM76.6）会设为 1。如果希望检测随后的 PTO 溢出位（SM66.6 或 SM76.6），必须在检测到 PTO 溢出位（SM66.6 或 SM76.6）为 1 后马上以手动方式清零该位，否则不能再次检测。每当 CPU 转换至 RUN（运行）模式时自动将该位清零。

注意：当载入新脉冲数（SMD72 或 SMD82）、脉冲宽度（SMW70 或 SMW80）或周期时间（SMW68 或 SMW78）时，在执行 PLS 指令之前，还需要在控制寄存器中设置适当的允许更新值。对于多段脉冲链操作，在执行 PLS 指令之前，还必须载入轮廓表的起始偏移量（SMW168 或 SMW178）和轮廓表个数。

PWM 的同步更新和异步更新如下描述：

同步更新：如果不要求更改时基，即可以执行同步更新。执行同步更新时，波形特征的变化发生在循环边缘，提供平滑转换。

异步更新：PWM 操作一般都是使用同步更新，即只改变脉冲宽度，不改变脉冲周期；但是一定要改变时基的话，就只能使用异步更新。异步更新使 PTO/PWM 发生器被立即暂停，强迫与 PWM 波形异步一直达到指定的时基，这样可能造成控制设备状态暂时不稳从而引起抖动。正是由于此原因，建议尽量使用同步更新，选择可用于所有预计周期数值的时基。

控制字节中的 PWM 更新位（SM67.4 或 SM77.4）指定更新类型（0 = 异步更新、1 = 同步更新），在执行 PLS 指令时激活改动。但注意，如果时基改变，则会发生异步更新，无论 PWM 更新位（SM67.4 或 SM77.4）的状态如何。

问 4：如何应用发 PWM 脉冲？

答：PWM 功能提供连续的脉冲输出，在脉冲输出的同时，如果需要，允许改变脉冲周期及脉冲宽度。脉冲周期及脉冲宽度的时基可以指定为 μs（微秒）或 ms（毫秒），如果需要在脉冲输出允许改变时基。

脉冲周期的范围为 10 ~ 65535μs 或 2 ~ 65535ms。

脉冲宽度的范围从 0 ~ 65535μs 或 0 ~ 65535ms。

如果“脉冲宽度时间≥脉冲周期”数值，工作循环为 100%，输出长期为“1”。

如果“脉冲宽度时间 = 0”，工作循环为 0%，输出长期为“0”。

如果脉冲周期小于 2 个时间单位，脉冲周期自动取默认值为两个时间单位。

一般初始化 PWM 的步骤如下是，使 SMB67 = 16#D3（选择微秒）或 16#DB（选择毫秒）。控制字中选择启用 PTO/PWM 功能、选择 PWM 操作、设置更新脉冲宽度和脉冲周期数值以及选择时基（微秒或毫秒）。

一般使用一个子程序初始化 PWM。在主程序需要时调用初始化 PWM 的子程序。PWM 应用程序如图 15-3 ~ 图 15-5 所示。

调试发 PWM 脉冲程序：

首先，把程序下载到 CPU 并至 RUN 模式。

发周期是 2000ms 和宽度是 1000ms 的脉冲。接通 I0.2 后，可以看到 Q0.0 发出指定参数的脉冲。

发周期是 30000μs 和宽度是 10000μs 的脉冲。接通 I0.3 后，可以看到 Q0.0 发出指定参数的脉冲。

停止发脉冲。接通 I0.4 后，脉冲停止了。

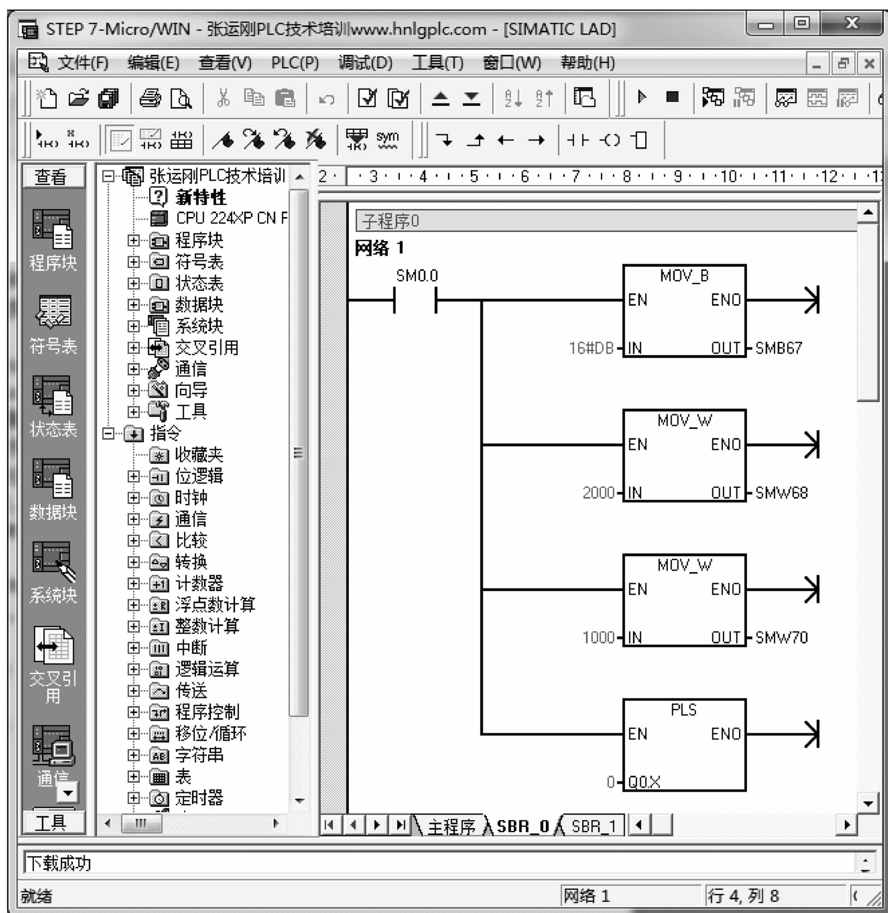


图 15-3　初始化 PWM 子程序 0

问 5：如何应用发单段 PTO 脉冲?

答： 单段 PTO 功能提供预先给定脉冲周期和脉冲个数的连续脉冲输出，在脉冲输出的过程中改变脉冲周期及脉冲个数是不允许的。脉冲周期及脉冲宽度的时基可以指定为 μs（微秒）或 ms（毫秒），在脉冲输出过程中也不允许改变时基。

当当前脉冲完成时，允许立即输出新的脉冲，这样就保证了随后的输出脉冲的连续性。

脉冲周期时间的可能范围为 10 ~ 65535μs 或 2 ~ 65535ms。

脉冲个数的可能范围为 1 ~ 4294967295 个脉冲。

如果把脉冲周期时间指定为基数微秒或毫秒（例如 35ms）会引起工作循环的失真。

在使用 PTO 指令当设定脉冲个数为“0”时，实际输出脉冲数为“1”个脉冲。

判断 PTO 输出脉冲串完成与否有两种方法：

1）监控特殊继电器（SM66.7 或 SM76.7）中的 PTO 空闲位状态可以知道编程脉冲串已完成。

2）可在单段脉冲串（或多段脉冲串）完成时激活中断程序。

在初始化子程序中按照以下步骤建立控制逻辑配置脉冲输出，然后从主程序调用初始化子程序。下面以 Q0.0 发脉冲为例说明：

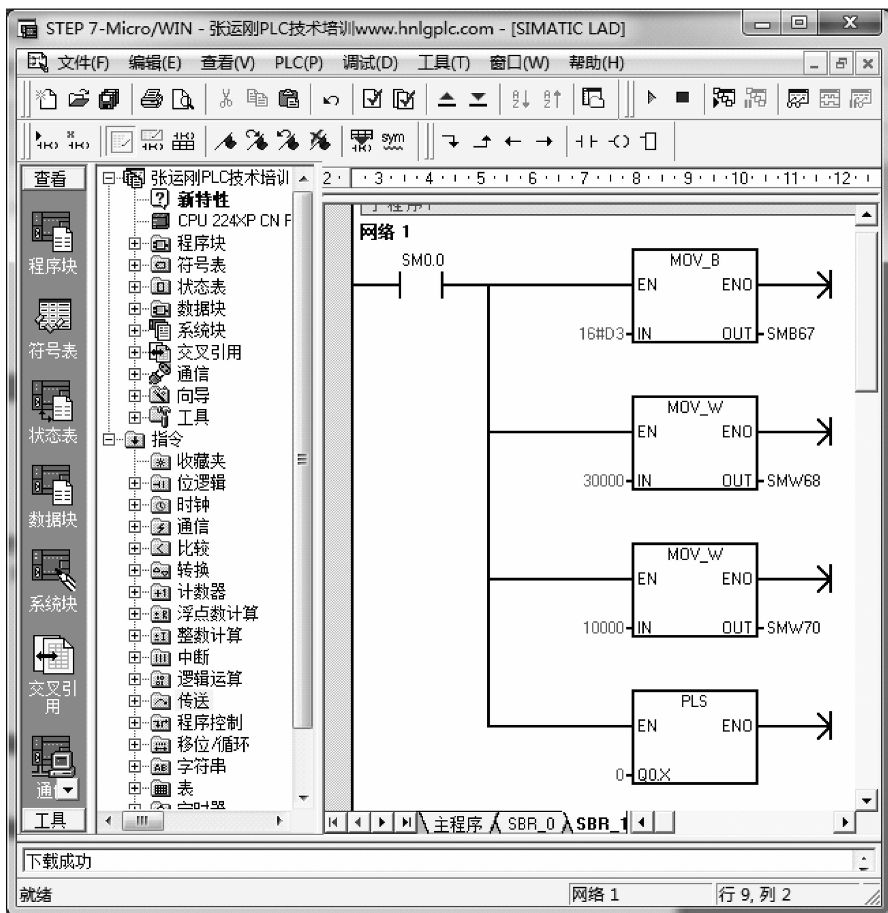


图 15-4　初始化 PWM 子程序 1

1）将 16#85（选择微秒）或 16#8D（选择毫秒）载入控制字节 SMB67。

2）把需要的初始脉冲周期载入 SMW68 中，把需要的脉冲数载入 SMD72 中。

3）如果希望在脉冲串输出完成后立即执行相关功能，可以将脉冲串完成事件（中断号 19）附加于中断程序，为中断编程，使用 ATCH 指令指定关联中断程序并执行 ENI 允许开中断指令。

4）执行 PLS 指令，使 S7-200 激活 PTO 脉冲发生器编程。

一般使用一个子程序初始化 PTO。在主程序需要时调用初始化 PTO 的子程序。单段 PTO 应用程序如图 15-6 ~ 图 15-10 所示。

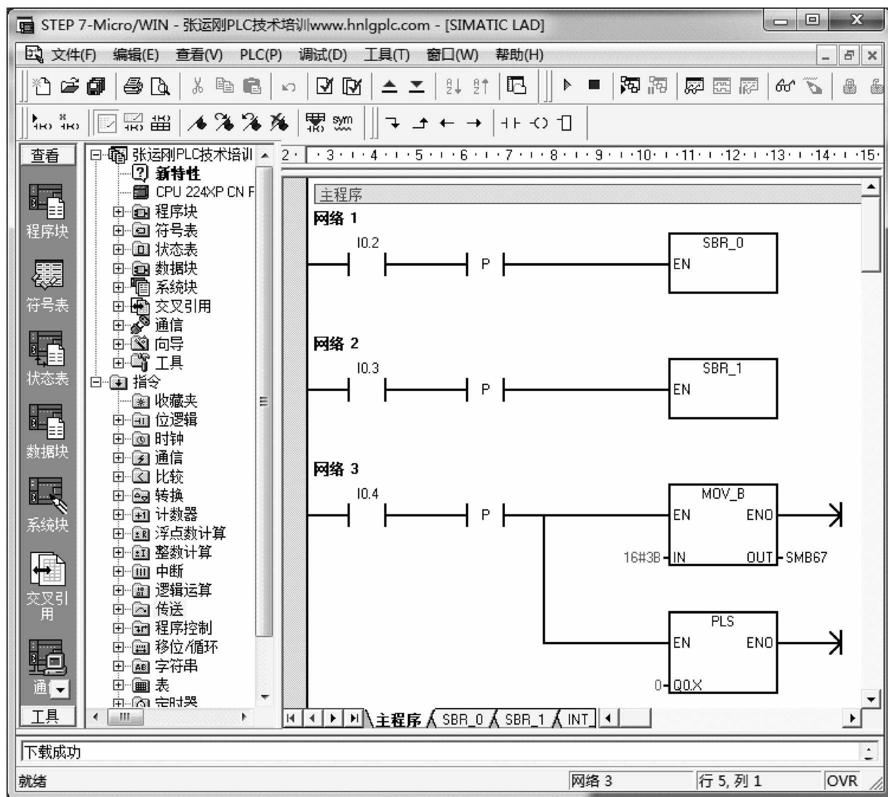


图 15-5　PWM 应用主程序

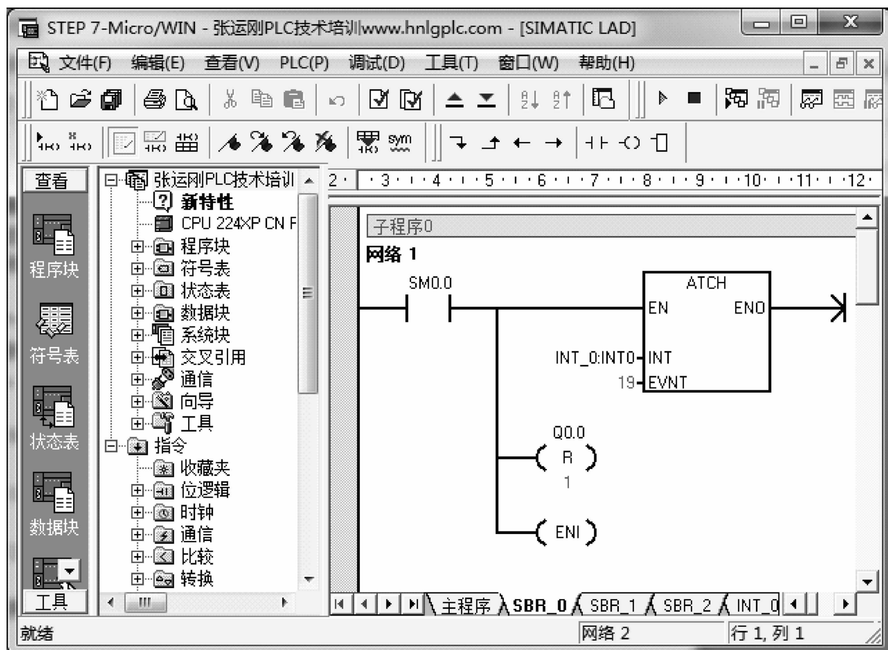


图 15-6　单段 PTO 子程序 0

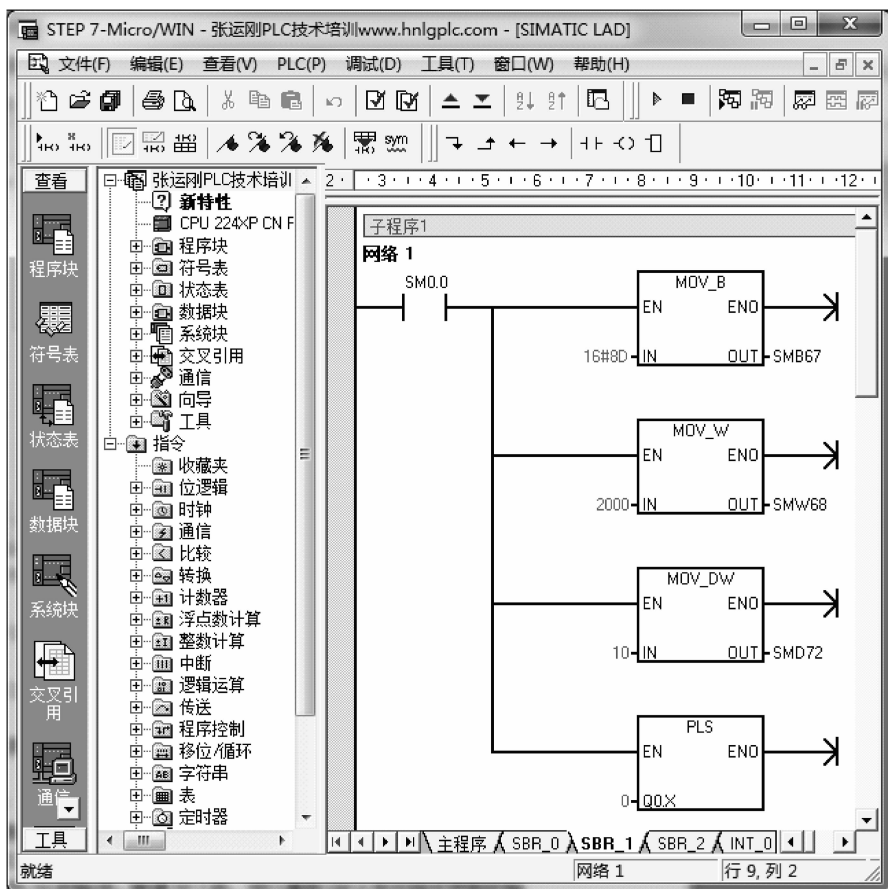


图 15-7　单段 PTO 子程序 1

调试发单段 PTO 脉冲程序：

首先，把程序下载到 CPU 并执行 RUN 模式。

发周期是 2000ms 和个数为 10 的脉冲。接通 I0.2 后，可以看到 Q0.0 发出指定参数的脉冲，脉冲发完后看到 Q0.1 的状态会变化。

发周期是 100μs 和个数为 1000 的脉冲。接通 I0.3 后，可以看到 Q0.0 发出指定参数的脉冲，脉冲发完后看到 Q0.1 的状态会变化。

停止发脉冲。当在发脉冲的过程中接通 I0.4 后，会马上停止发脉冲。

连续发两种参数的脉冲。接通 I0.2 后，马上接通 I0.3，可以看到先发周期是 2000ms 和个数为 10 的脉冲，然后紧接发周期是 100μs 和个数为 1000 的脉冲。

问 6：如何应用发多段 PTO 脉冲？

答：使用多段 PTO 操作时按照驱动轮廓图的要求，生成轮廓表，使 PTO 按照指定参数完成复杂的运动轨迹驱动。

图 15-11 所示是步进电动机典型的驱动轮廓图，包括起动加速、匀速运行和降速停止 3 个阶段。第一阶段是步进电动机加速起动，第二阶段是步进电动机匀速过程，第三阶段是步进电动机减速停止。

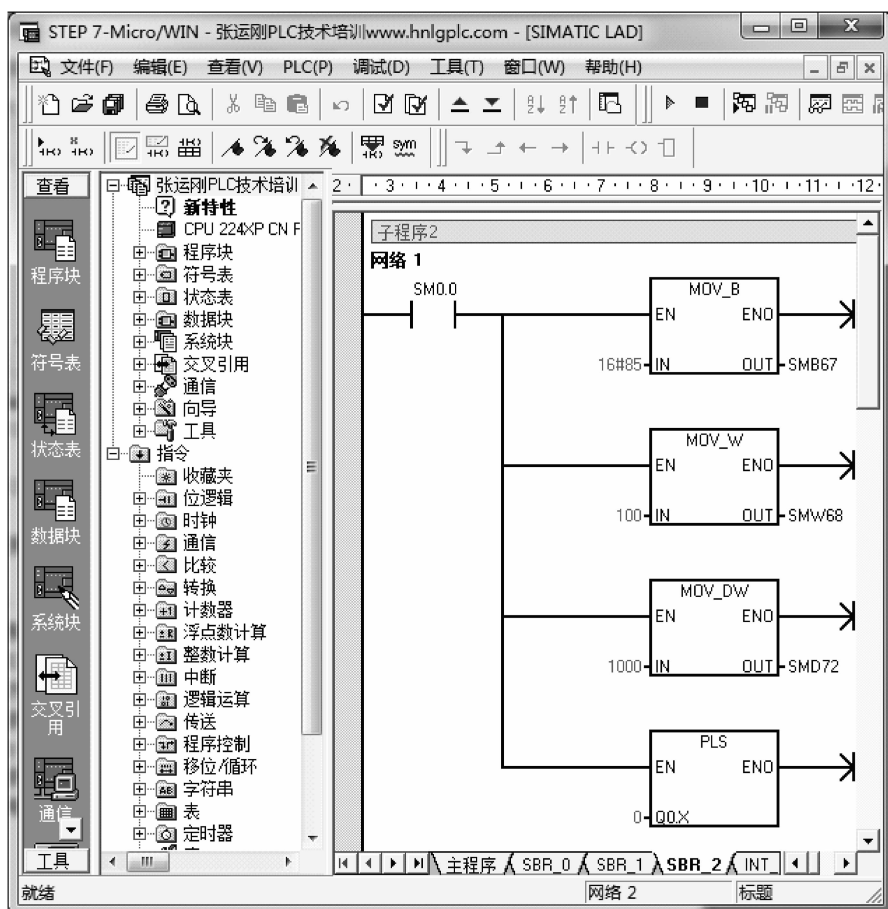


图 15-8　单段 PTO 子程序 2



图 15-9　单段 PTO 中断程序 0

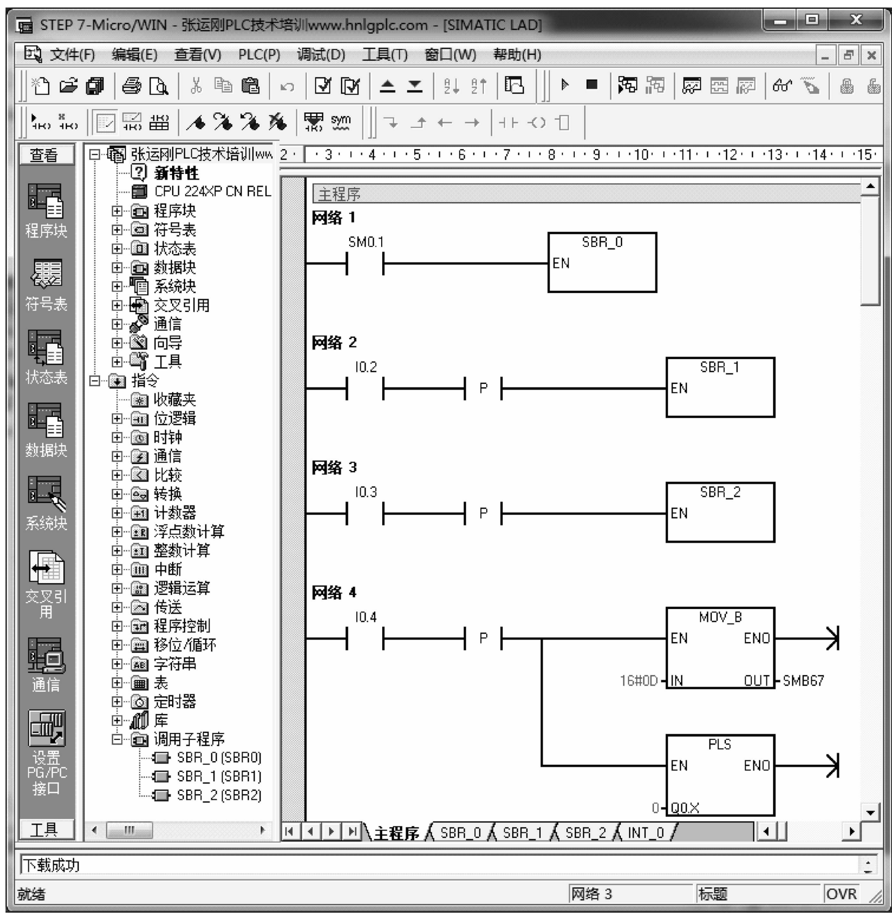


图 15-10　单段 PTO 主程序

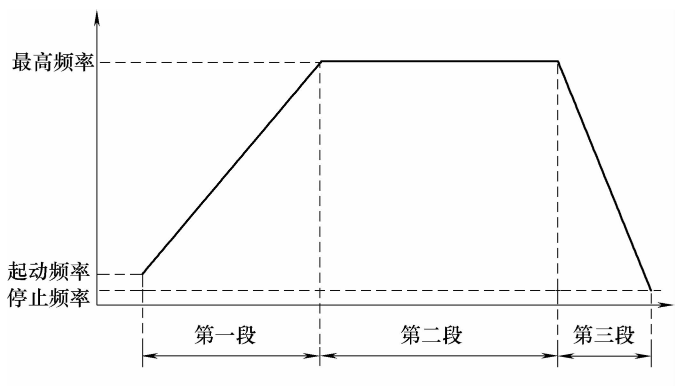


图 15-11　3 段 PTO 典型应用

使用轮廓表来表达多段 PTO 时，表达各段参数特征时，轮廓表遵循一定的规律，见表 15-2。

表 15-2　轮廓表规律

轮廓表字节地址偏移	轮廓段		解　　释
0	B	总段数	多段 PTO 总段数(范围:1 ~ 255)
1	W	第一段	该段初始周期(范围:2 ~ 65535)
3	I		脉冲周期增量(范围:－32768 ~ ＋32767)
5	DW		该段脉冲总数(范围:1 ~ 4294967295)
9	W	第二段	该段初始周期(范围:2 ~ 65535)
11	I		脉冲周期增量(范围:－32768 ~ ＋32767)
13	DW		该段脉冲总数(范围:1 ~ 4294967295)
17	W	第三段	该段初始周期(范围:2 ~ 65535)
19	I		脉冲周期增量(范围:－32768 ~ ＋32767)
21	DW		该段脉冲总数(范围:1 ~ 4294967295)
…		…	…

假设一个实际应用中要求：起动频率为 2kHz、停止频率为 1kHz，最大脉冲频率为 10kHz，要求 8000 个脉冲完成第一段到第三段的所有过程。

因为 PTO 轮廓表使用脉冲周期来表达而不是频率，需要将指定频率数值转换成脉冲周期数值。因此，起动频率为 2kHz 折算成脉冲周期是 500μs、停止频率为 1kHz 折算成脉冲周期为 1000μs，最大脉冲频率为 10kHz 折算成脉冲周期为 100μs。

在驱动轮廓图的加速部分，要求大约 400 个脉冲时达到最大脉冲频率，所以脉冲增量为“－1”。在驱动轮廓图的减速部分，要求大约 450 个脉冲时达到停止脉冲频率，所以要求脉冲增量为“2”。

如果轮廓表的开始位置是 VB100，那么表达这个应用的具体轮廓参数见表 15-3。

表 15-3　实际应用特例轮廓参数表

轮廓段	轮廓表字节地址偏移	数值	解　　释
开始位置	VB100	3	多段 PTO 总段数(范围:1 ~ 255)
第一段	VW101	500μs	该段初始周期(范围:2 ~ 65535)
	VW103	－1	脉冲周期增量(范围:－32768 ~ ＋32767)
	VD105	400	该段脉冲总数(范围:1 ~ 4294967295)
第二段	VW109	100μs	该段初始周期(范围:2 ~ 65535)
	VW111	0	脉冲周期增量(范围:－32768 ~ ＋32767)
	VD113	7150	该段脉冲总数(范围:1 ~ 4294967295)
第三段	VW117	100μs	该段初始周期(范围:2 ~ 65535)
	VW119	2	脉冲周期增量(范围:－32768 ~ ＋32767)
	VD121	450	该段脉冲总数(范围:1 ~ 4294967295)

一般使用一个子程序初始化 PTO。在主程序需要时调用初始化 PTO 的子程序。多段 PTO 应用程序如图 15-12 ~ 图 15-15 所示。

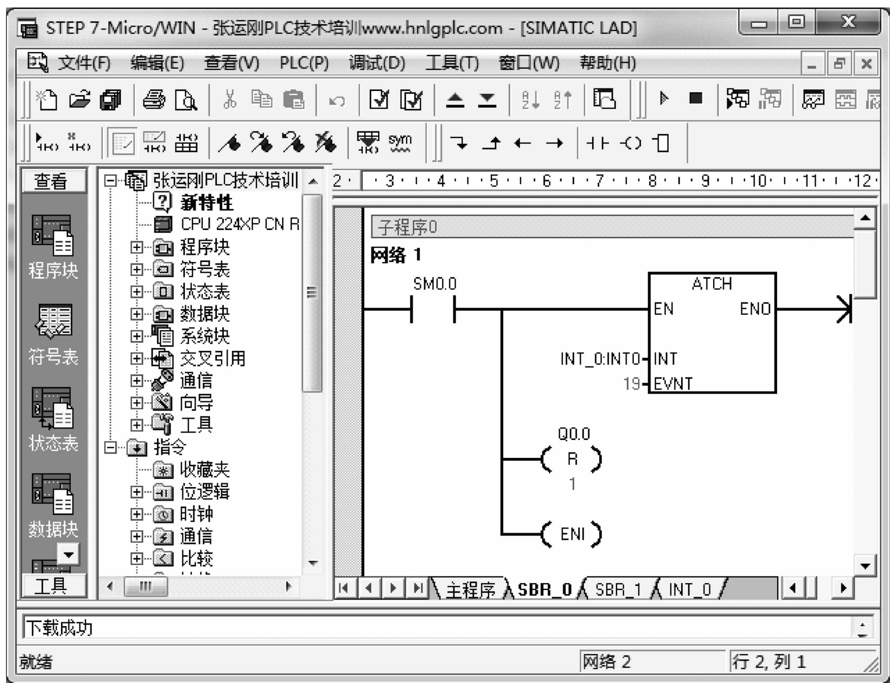


图 15-12　多段 PTO 子程序 0

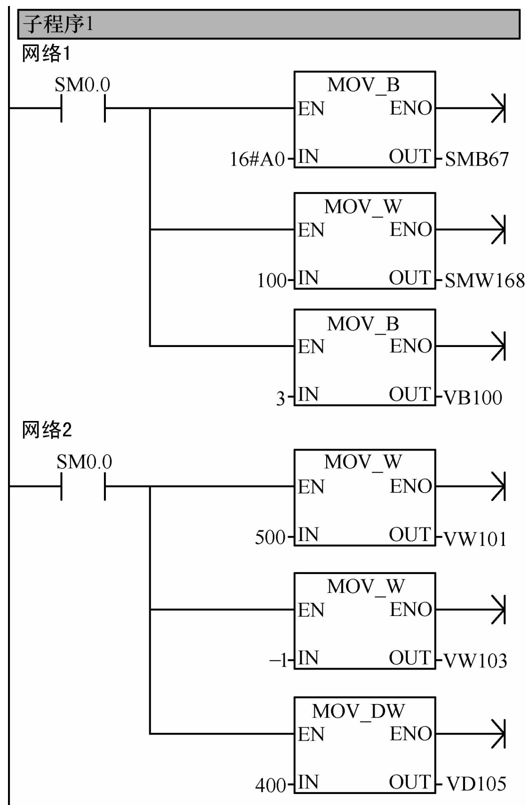


图 15-13　多段 PTO 子程序 1

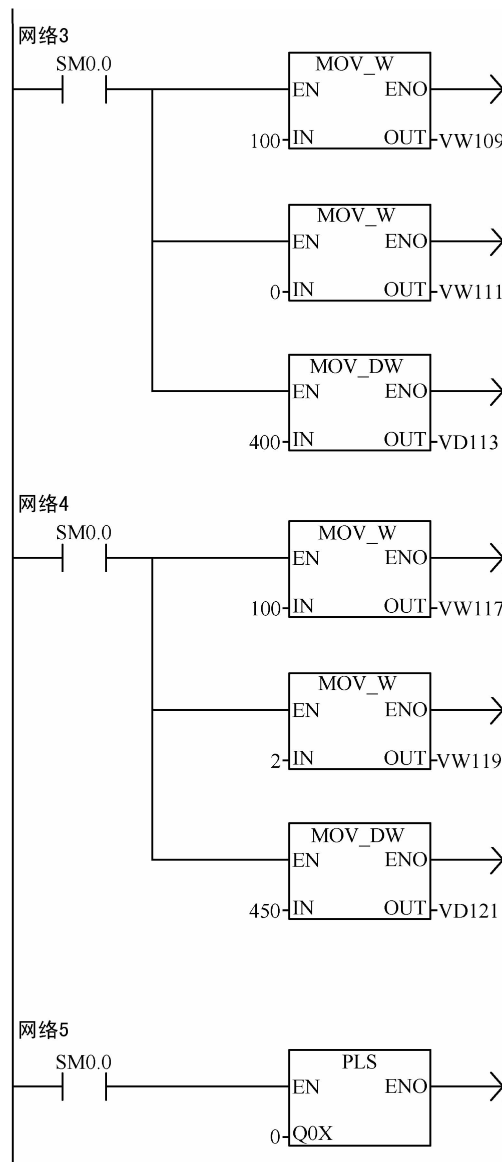


图 15-13　多段 PTO 子程序 1（续）

调试发多段 PTO 脉冲程序：

首先，把程序下载到 CPU 并执行 RUN 模式。

发多段 PTO 脉冲。接通 I0.2 后，看到 Q0.0 发出预先给定的参数发脉冲，脉冲发完后看到 Q0.1 的状态会变化。

停止发脉冲。在发脉冲过程中，接通 I0.4 后，脉冲停止了。

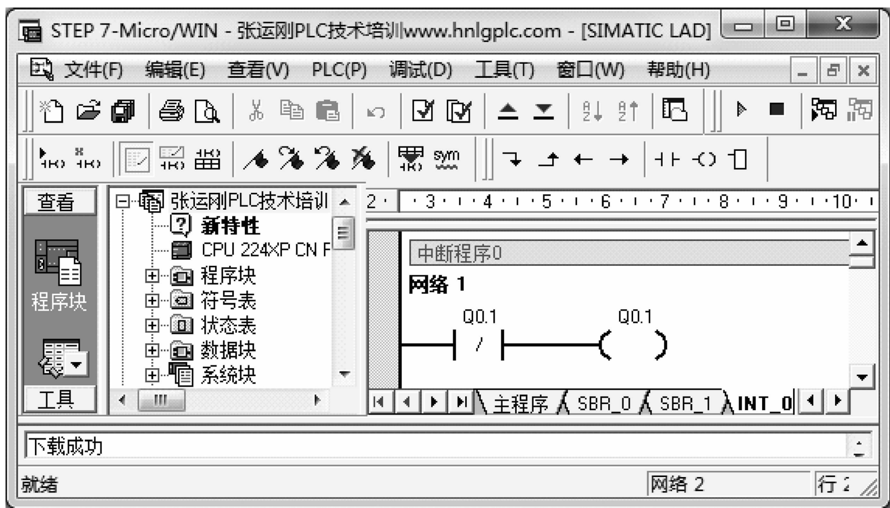


图 15-14　多段 PTO 中断程序 0

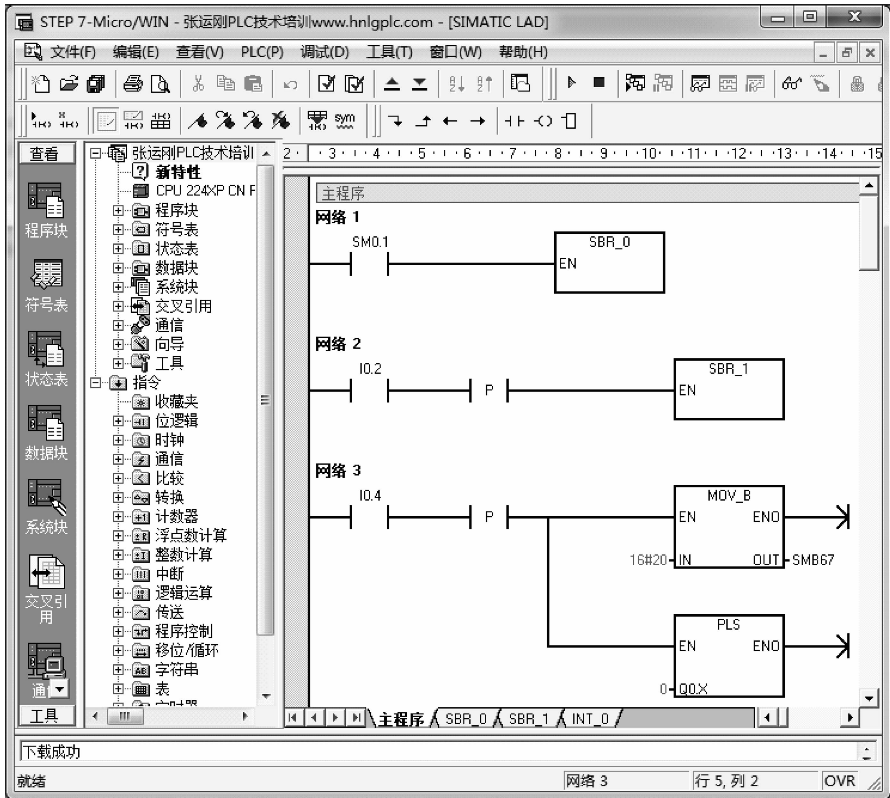


图 15-15　多段 PTO 主程序

第 16 章　累加器和指针

问：S7-200 CPU 有几个累加器？ 累加器如何使用？ S7-200 CPU 在程序中有几种寻址方式？ 间接寻址俗称为指针寻址，如何使用？

问 1：S7-200 CPU 有几个累加器？

答：S7-200 CPU 有 4 个累加器，分别是 AC0、AC1、AC2 和 AC3。

问 2：累加器如何使用？

答：累加器在编程时可以按字节（8 位）、字（16 位）或双字（32 位）来访问，访问的长度取决于执行的指令。

累加器像一般的存储器一样，可以进行读写操作，也可以向子程序传递参数，或者从子程序返回参数以及用来暂存程序中的中间结果。但是累加器没有停电保持性能，CPU 在每次 RUN 前会复位所有累加器。

如果用户的程序允许开中断，那么中断程序随时会执行，考虑到累加器可以同时在主程序和中断程序间共用而不受影响，系统在执行中断程序前自动对累加器的状态保存，在从中断程序出来之前恢复累加器的状态。这样可避免因分支至中断程序和从中断程序分支而导致的主程序累加器状态的混乱。

如果在主程序和子程序同时使用累加器时而做到互不影响，那么进入执行子程序之前需要将累加器的数据保护起来，在出来之前恢复就可以了。同样在子程序也使用同样方法把累加器数据保护起来即可。

使用累加器存储中间结果方法计算“［VB10（8 位无符号数）＋VW11（16 位整数）］÷VD13（32 位小数）”这个数学题目，把最后结果保存在 VW17 中，实现的程序如图 16-1 所示。

问 3：S7-200 CPU 在程序中有几种寻址方式？

答：寻址方式有 3 种：立即寻址、直接寻址和间接寻址。立即寻址和直接寻址如图 16-2 所示，间接寻址如图 16-3 所示。

问 4：间接寻址俗称为指针寻址，如何使用？

答：指针变量是 32 位的存储器，可以是 VD、LD 或者是 AC，比如*VD、*LD 和*AC。

指针变量见表 16-1。指向的软元件类型见表 16-2。

使用指针三部曲：一是建立指针基准，二是应用指针，三是指针偏移。

试运算从 VW100 开始连续 *N* 个数的平均值。*N* 个数由 VW98 给定，运算结果放置在 VW96 里面。实现的程序如图 16-4 所示。

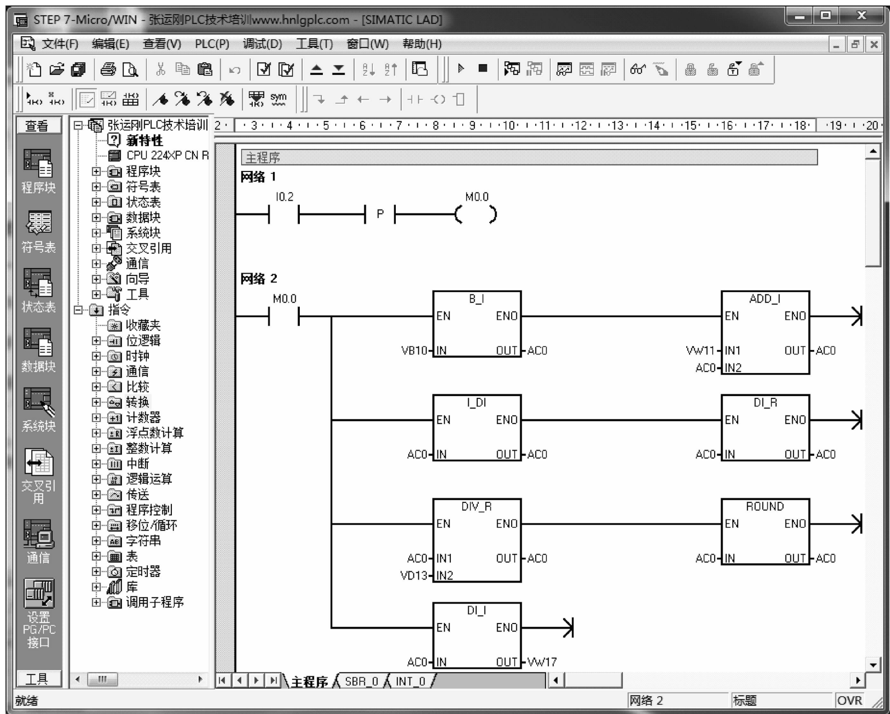


图 16-1　累加器应用程序 1

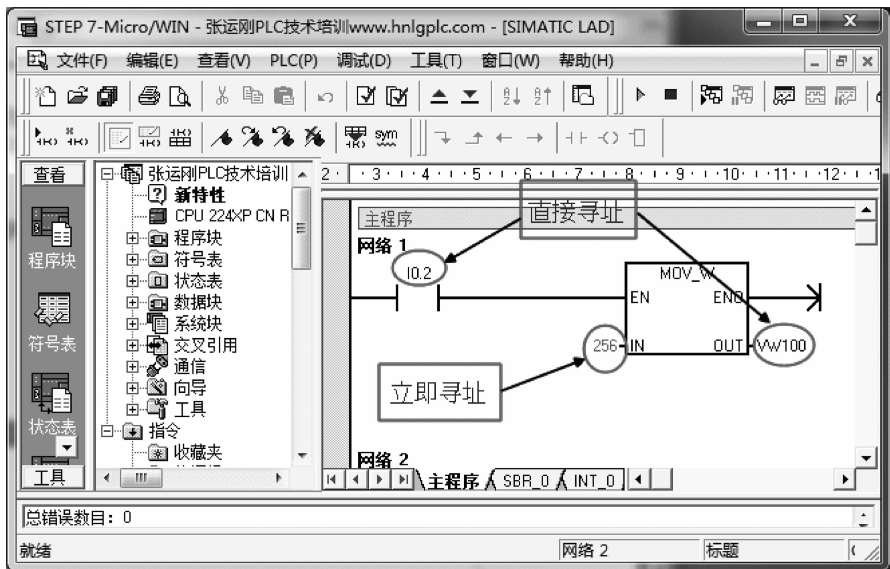


图 16-2　立即寻址和直接寻址程序

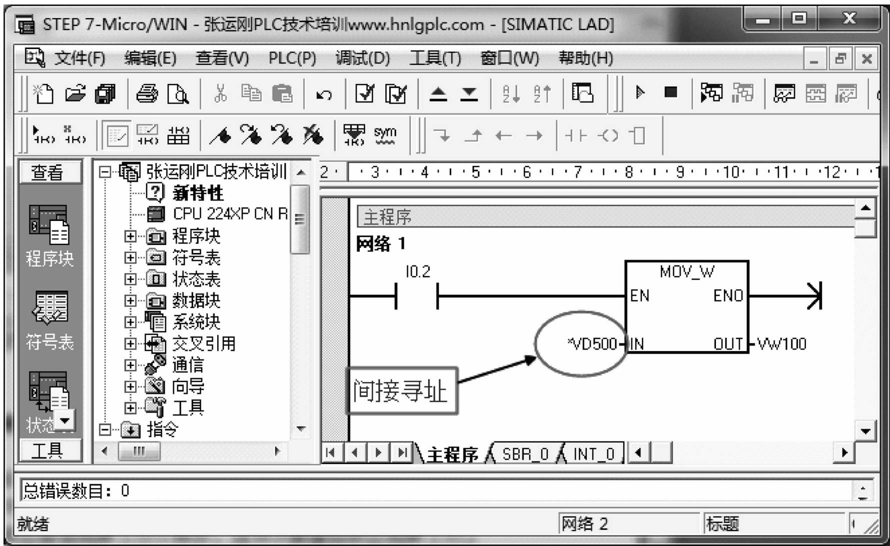


图 16-3　间接寻址程序

表 16-1　指针变量

				软元件类型												字节地址															

表 16-2　指针变量软元件类型

二进制数值				软元件
0	0	0	0	I
0	0	0	1	Q
0	0	1	0	M
0	0	1	1	S
0	1	0	0	SM
1	0	0	0	V
1	0	0	1	T
1	0	1	0	C

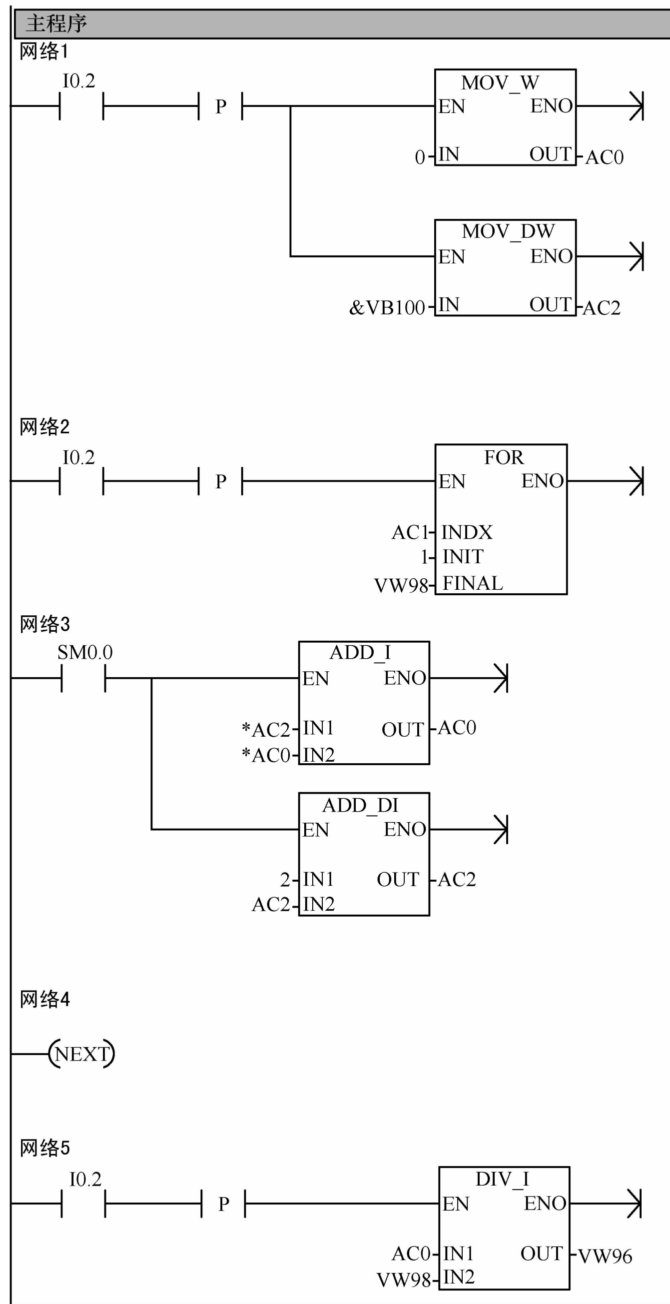


图 16-4　间接寻址应用程序

在应用指针时，注意超出软元件范围错误，如图 16-5 所示的程序中，当接通 I0. 2 后，再次接通 I0. 3 将会出现访问软元件 M 超出范围错误。

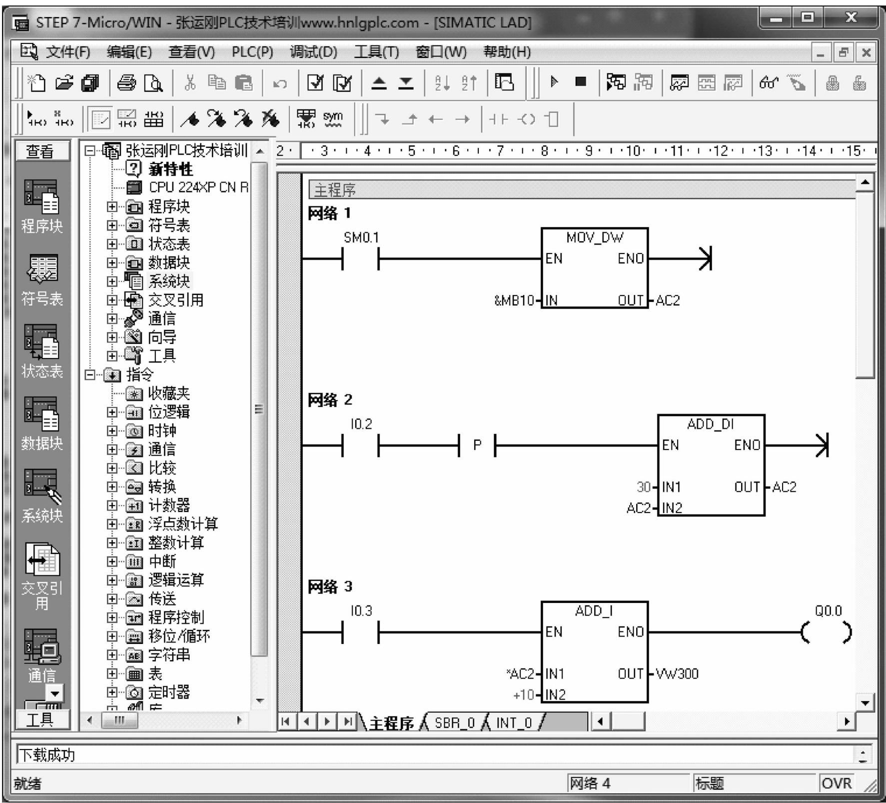


图 16-5　指针应用程序例

第 17 章　扩展模块与模拟量

17.1　模块和地址

问：S7-200 的 CPU 有哪些？
S7-200 的 CPU 技术规范怎样？
常用的数字量扩展模块有哪些？
常用的模拟量扩展模块有哪些？
S7-200 的特殊功能模块有哪些？
扩展模块的地址分配规律是什么？

问 1：S7-200 的 CPU 有哪些？

答：S7-200CPU 见表 17-1。

表 17-1　S7-200CPU 列表

CPU 系列号	产品外形图	描　述	订　货　号
CPU221		DC/DC/DC;6 点输入/4 点输出	6ES7 211-0AA23-0XB0
		AC/DC/继电器;6 点输入/4 点输出	6ES7 211-0BA23-0XB0
CPU222		DC/DC/DC;8 点输入/6 点输出	6ES7 212-1AB23-0XB0
		AC/DC/继电器;8 点输入/6 点输出	6ES7 212-1BB23-0XB0
CPU224		DC/DC/DC;14 点输入/10 点输出	6ES7 214-1AD23-0XB0
		AC/DC/继电器;14 点输入/10 点输出	6ES7 214-1BD23-0XB0
CPU224XP		DC/DC/DC;14 点输入/10 点输出;2 输入/1 输出共 3 个模拟量 I/O 点	6ES7 214-2AD23-0XB0
		AC/DC/继电器;14 点输入/10 点输出;2 输入/1 输出共 3 个模拟量 I/O 点	6ES7 214-2BD23-0XB0

（续）

CPU 系列号	产品外形图	描　　述	订　货　号
CPU226		DC/DC/DC;24 点输入/16 点晶体管输出	6ES7 216-2AD23-0XB0
		AC/DC/继电器;24 点输入/16 点输出	6ES7 216-2BD23-0XB0
CPU226XM		DC/DC/DC;24 点输入/16 点晶体管输出	6ES7 216-2AF22-0XB0
		AC/DC/继电器;24 点输入/16 点输出	6ES7 216-2BF22-0XB0

问 2：S7-200 的 CPU 技术规范怎样？

答：S7-200 的 CPU 技术规范见表 17-2。

表 17-2　CPU 的技术规范表

技术规范	CPU221	CPU222	CPU224	CPU224XP	CPU226
存储器特性					
存储器用户程序大小 ● 运行模式下编辑 ● 非运行模式下编辑	4096 字节 4096 字节	4096 字节 4096 字节	8192 字节 12288 字节	12288 字节 16384 字节	16384 字节 24576 字节
用户数据	2048 字节	2048 字节	8192 字节	10240 字节	10240 字节
掉电保持（超级电容）	50h/典型值（40°C 时最少 8h）		100h/典型值（40°C 时最少 70h）		
（可选电池）	200 天/典型值	200 天/典型值	200 天/典型值	200 天/典型值	200 天/典型值
本机 I/O 特性					
本机数字量 I/O	6 输入/4 输出	8 输入/6 输出	14 输入/10 输出	14 输入/10 输出	24 输入/16 输出
本机模拟量 I/O	无	无	无	2 输入/1 输出	无
数字 I/O 映像区	256（128 输入/128 输出）				
模拟 I/O 映像区	无	32（16 输入/16 输出）	64（32 输入/32 输出）		
允许最大的扩展 I/O 模块	无	2 个模块	7 个模块	7 个模块	7 个模块
允许最大的智能模块	无	2 个模块	7 个模块	7 个模块	7 个模块
脉冲捕捉输入	6	8	14	14	24
高速计数器总数 ● 单相计数器 ● 两相计数器	总共 4 个计数器 4 个 30kHz 2 个 20kHz	总共 4 个计数器 4 个 30kHz 2 个 20kHz	总共 6 个计数器 6 个 30kHz 4 个 20kHz	总共 6 个计数器 4 个 30kHz 2 个 200kHz 3 个 20kHz 1 个 100kHz	总共 6 个计数器 6 个 30kHz 4 个 20kHz
脉冲输出	2 个 20kHz（仅限于 DC 输出）	2 个 20kHz（仅限于 DC 输出）	2 个 20kHz（仅限于 DC 输出）	2 个 100kHz（仅限于 DC 输出）	2 个 20kHz（仅限于 DC 输出）

（续）

技术规范	CPU221	CPU222	CPU224	CPU224XP	CPU226
常规特性					
定时器总数	256 定时器；4 个定时器（1ms）；16 定时器（10ms）；236 定时器（100ms）				
计数器总数	256（由超级电容或电池备份）				
内部存储器位掉电保存	256（由超级电容或电池备份） 112（存储在 EEPROM）				
时间中断	2 个 1ms 分辨率				
边沿中断	4 个上升沿和/或 4 个下降沿				
模拟电位器模拟电位器 1 个 8 位分辨率	1 个 8 位分辨率	1 个 8 位分辨率	2 个 8 位分辨率	2 个 8 位分辨率	2 个 8 位分辨率
布尔量运算执行时间	0.22μs 每条指令				
实时时钟	可选卡件	可选卡件	内置	内置	内置
卡件选项	存储器、电池和实时时钟	存储器、电池和实时时钟	存储卡和电池卡	存储卡和电池卡	存储卡和电池卡
集成的通信功能					
接口	1 个 RS-485 接口	1 个 RS-485 接口	1 个 RS-485 接口	2 个 RS-485 口	2 个 RS-485 口
PPI，DP/T 波特率	9.6、19.2、187.5kbit/s				
自由口波特率	1.2 ~ 115.2kbit/s				
每段最大电缆长度	使用隔离的中继器：187.5kbit/s 可达 1000m，38.4kbit/s 可达 1200m 未使用隔离中继器：50m				
最大站点数	每段 32 个站，每个网络 126 个站				
最大主站数	32				
点到点（PPI 主站模式）	是（NETR/NETW）				
MPI 连接	共 4 个，2 个保留（1 个给 PG，1 个给 OP）				

问 3：常用的数字量扩展模块有哪些？

答：常用的数字量模块见表 17-3。

表 17-3　数字量模块列表

系列号	类　　别	产品外形图	描　　述	订　货　号
EM221	输入扩展模块		8 点输入，24V DC	6ES7 221-1BF22-0XA0
			8 点输入，120/230VAC	6ES7 221-1EF22-0XA0
			16 点输入，24VDC	6ES7 221-1BH22-0XA0

（续）

系列号	类　　别	产品外形图	描　　述	订　货　号
EM222	输出扩展模块		4 点输出,24VDC	6ES7 222-1BD22-0XA0
			4 点输出,继电器	6ES7 222-1HD22-0XA0
			8 点输出,24V DC	6ES7 222-1BF22-0XA0
			8 点输出,继电器	6ES7 222-1HF22-0XA0
			8 点输出,120/230VAC	6ES7 222-1EF22-0XA0
EM223	输入/输出扩展模块		4 点输入,24V DC 4 点输出,24V DC	6ES7 223-1BF22-0XA0
			4 点输入,24V DC 4 点输出,继电器	6ES7 223-1HF22-0XA0
			8 点输入,24V DC 8 点输出,24V DC	6ES7 223-1BH22-0XA0
			8 点输入,24V DC 8 点输出,继电器	6ES7 223-1PH22-0XA0
			16 点输入,24V DC 16 点输出,24V DC	6ES7 223-1BL22-0XA0
			16 点输入,24V DC 16 点输出,继电器	6ES7 223-1PL22-0XA0

问 4：常用的模拟量扩展模块有哪些?

答：常用的模拟量扩展模块见表 17-4。

表 17-4　模拟量模块列表

系列号	类别	产品外形图	描　　述	订　货　号
EM 231	模拟量输入扩展模块		4 点模拟量输入	6ES7 231-0HC22-0XA0

（续）

系列号	类别	产品外形图	描　　述	订　货　号
EM 232	模拟量输出扩展模块		2 点模拟量输出	6ES7 232-0HB22-0XA0
EM 235	模拟量输入/输出扩展模块		3 点模拟量输入 1 点模拟量输出	6ES7 235

问 5：S7-200 的特殊功能模块有哪些?

答： S7-200 的特殊功能模块见表 17-5。

表 17-5　特殊功能模块列表

系列号	类　　别	产品外形图片	描　　述	订　货　号
EM 231	热电偶(TC)输入扩展模块		4 点热电偶(TC)输入	6ES7 231-7PD22-0XA0

（续）

系列号	类　　别	产品外形图片	描　　述	订　货　号
EM 231	2热电阻（RTD）输入点		2点热电阻（RTD）输入	6ES7 231-7PB20-0XA0
EM253	位置定位控制的模块		其输出信号Q用作位控功能的逻辑控制信号，不能直接驱动现场任何执行控制器件	6ES7 253-1AA22-0XA0
EM241	调制解调器模块	EM241 Modem Module	用于将S7-200连接在电话网上 利用安装在S7-200系统中的EM241 Modem模块，通过电话网，与CPU进行远程通信	6ES7 241-1AA22-0XA0
EM 277	PROFIBUS DP模块		EM 277 PROFIBUS-DP模块，可以作为PROFIBUS-DP从站和MPI从站	6ES7 277-0AA22-0XA0

（续）

系列号	类　　别	产品外形图片	描　　述	订　货　号
CP243-1	工业以太网模块		SIMATIC S7-200 与工业以太网的连接模块	6GK7 243-1EX00-0XE0
CP 243-2	AS-i 模块		SIMATIC S7-200CPU 22x 的 AS-i 主站	6GK7 243-2AX01-0XA0

问 6：扩展模块的地址分配规律是什么？

答：数字量扩展模块的 I/O 地址分配是非常有规律的，都是按照字节为单位往扩展方向递增；扩展模拟量模块的地址分配也是非常有规律的，都是按照两个字为单位往扩展方向递增；有些特殊模块会占用 Q 字节地址。

比如 CPU222 扩展两个数字量扩展模块 EM221 和 EM222，CPU 基本单元占用 IB0 和 QB0，EM221 的地址是 IB1，EM222 的地址是 QB1，如图 17-1 所示。

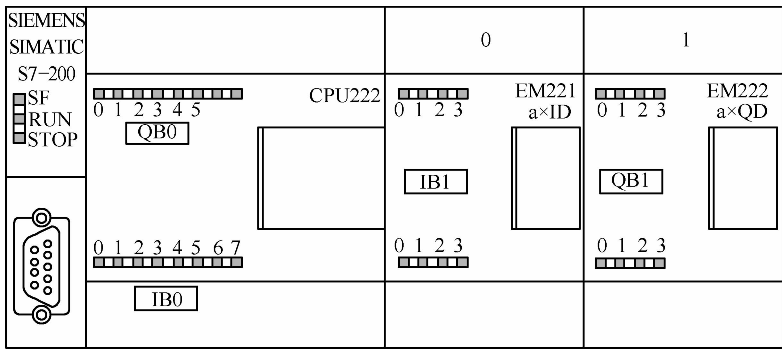


图 17-1　数字量扩展模块的 I/O 地址分配例 1

又比如 CPU224 扩展 3 个数字量扩展模块 EM221、EM222 和 EM223（16I/16Q），CPU 基本单元占用 IB0、IB1 和 QB0、QB1，EM223 的地址是 IB2、IB3 和 QB2、QB3，EM222 的地址是 QB4，EM221 的地址是 IB4，如图 17-2 所示。

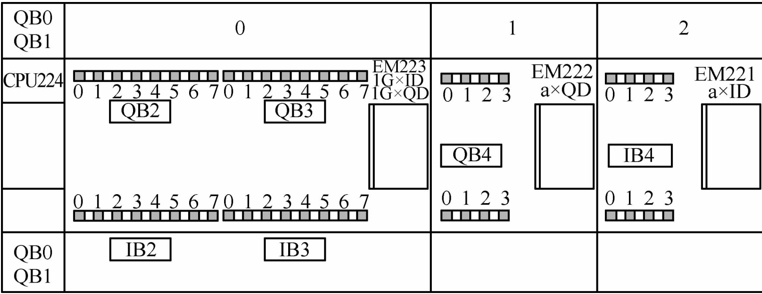


图 17-2　数字量扩展模块的 I/O 地址分配例 2

比如 CPU222 扩展两个模拟量扩展模块 EM235 和 EM232，EM235 的地址是 AIW0、AIW2、AIW4、AIW6 和 AQW0，EM232 的地址是 AQW4 和 AQW6，如图 17-3 所示。

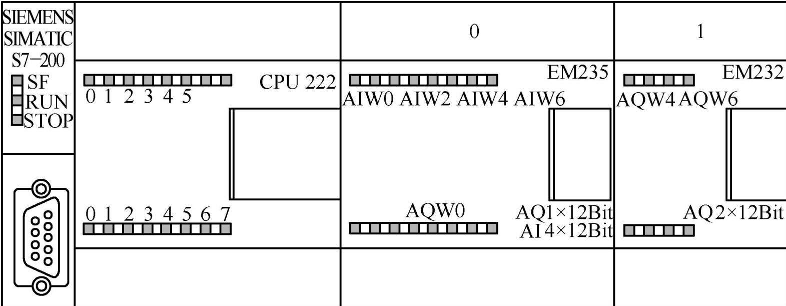


图 17-3　模拟量扩展模块的地址分配例 1

比如 CPU226 扩展两个模拟量扩展模块和两个数字量模块，模拟量模块的地址与数字量模块的地址分配互不干扰，是互相独立的，各模块的地址分配如图 17-4 所示。

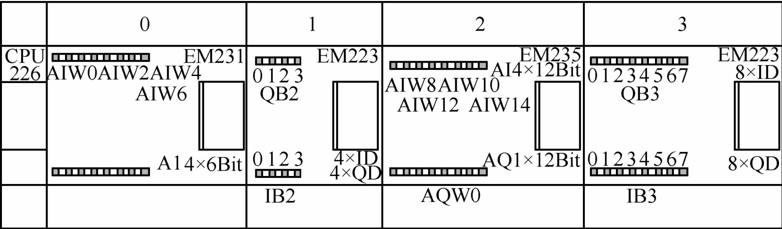


图 17-4　模拟量和数字量扩展模块的地址分配例 2

17.2　模拟量表示法

问：工业标准的模拟量是什么？
PLC 对模拟量处理流程是怎样的？
S7-200 的 CPU 怎样表达模拟值？

问1：工业标准的模拟量是什么？

答：工业标准的模拟量是：电流4～20mA，电压0～5V或0～10V。

问2：PLC对模拟量处理流程是怎样的？

答：工业现场有很多的模拟量都是非标准的，比如压力、温度、速度、流量、重量、黏度和长度等等。PLC无法直接采集这些模拟值，需要通过变送器把这些模拟值变成工业标准的模拟值后，CPU通过模拟量模块采集这些非标的模拟值。

S7-200PLC对模拟量输入流程如图17-5所示；对模拟量输出流程如图17-6所示。

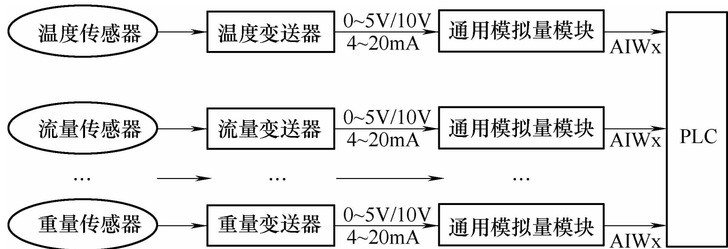


图17-5　模拟量输入流程

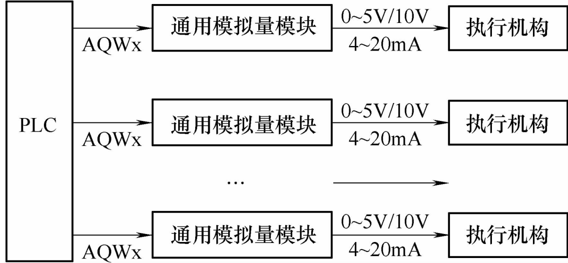


图17-6　模拟量输出流程

问3：S7-200的CPU怎样表达模拟量？

答：S7-200的常用模拟量模块位数有12位、13位、14位和16位的，其表达的模拟量范围和分辨率见表17-6所示，其二进制位中最高一位是符号位，0代表是正数，1代表是负数。

表17-6　S7-200模拟量表

位数	分辨率	模拟值十进制	模拟值二进制															
12	16	-32000～+32000	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0	0
13	8	-32000～+32000	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0
14	4	-32000～+32000	X	X	X	X	X	X	X	X	X	X	X	X	X	X	0	0
16	1	-32000～+32000	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

对于CPU224 XP CN CPU自带有两路模拟量输入和一路模拟量输出，两路模拟量输入分别是AIW0和AIW2，模拟量输出是AQW0。

现在使用CPU224 XP CN CPU自带模拟量编程控制一个温度控制系统，通过AIW2检测温度，通过AQW0控制风路的风门开度，从而控制着温度。系统控制系统如图17-7所示。实现程序如图17-8和图17-10所示。VW506是每隔0.1s检测到的实际温度值，单位是0.1℃；VW508是代表控制风门开度的比例阀动态参数，VW508=0～32000时代表风门打开0～100%的开度。

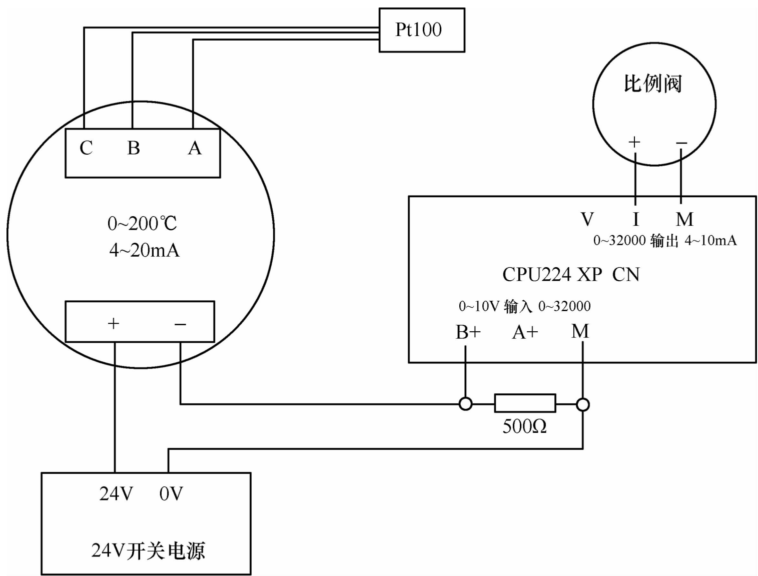


图 17-7 温度控制系统示意图

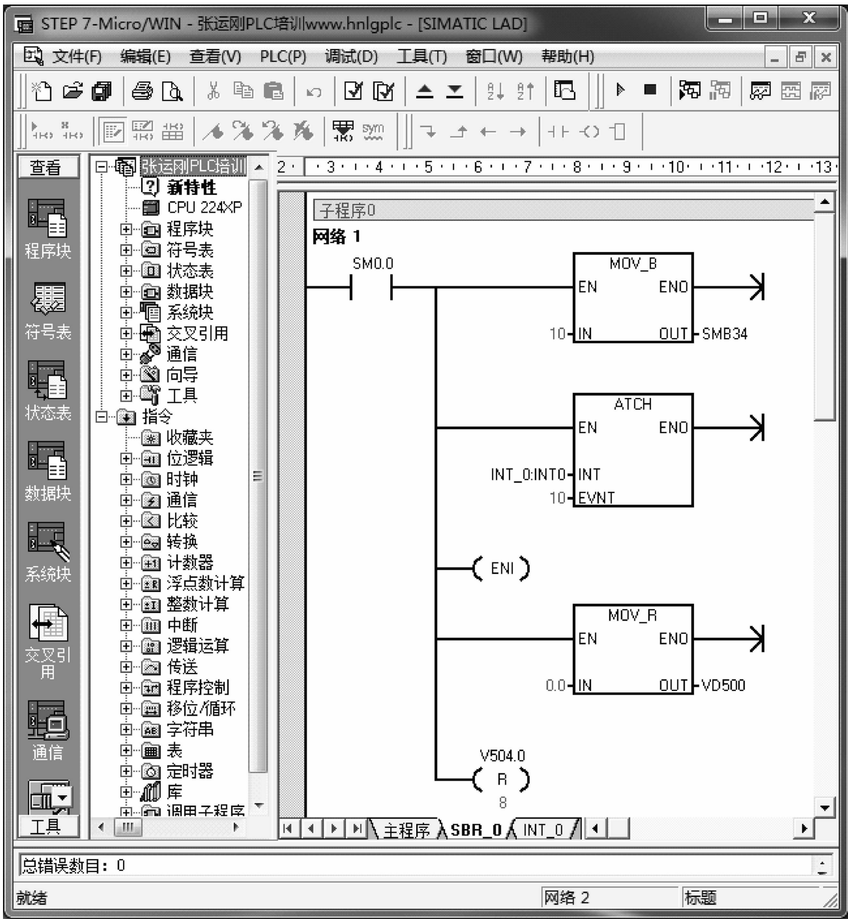


图 17-8 初始化子程序 0

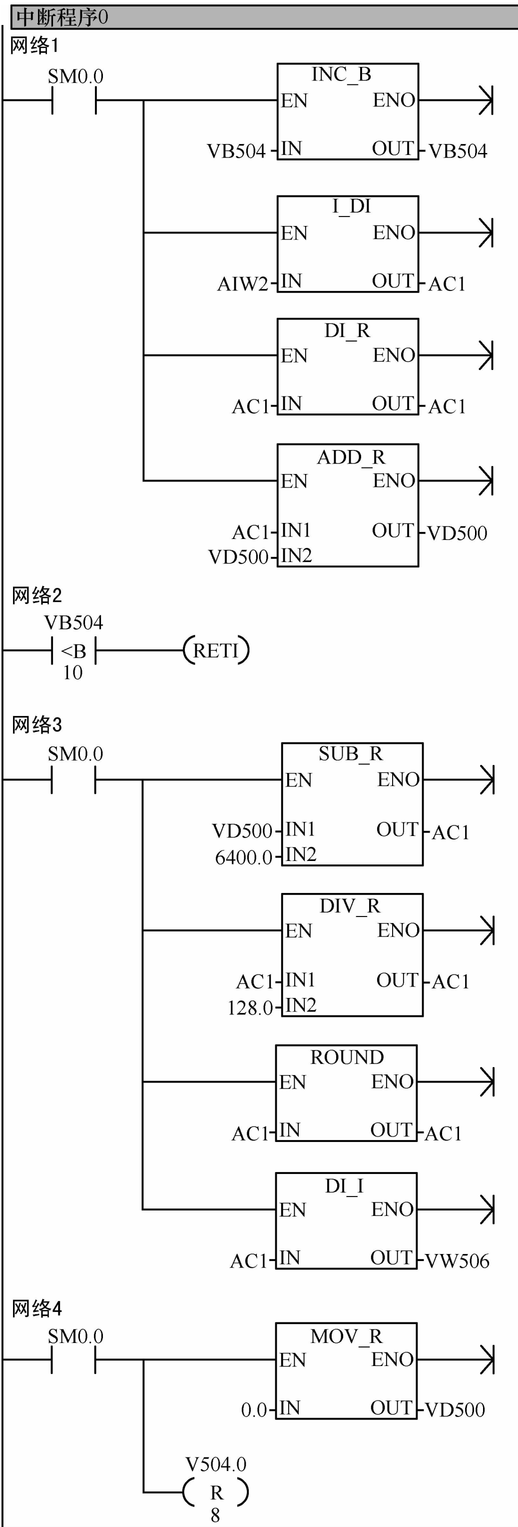

中断程序0
网络1
SM0.0
INC_B
EN
ENO
VB504
IN
OUT
VB504
I_DI
EN
ENO
AIW2
IN
OUT
AC1
DI_R
EN
ENO
AC1
IN
OUT
AC1
ADD_R
EN
ENO
AC1
IN1
OUT
VD500
VD500
IN2
网络2
VB504
<B
10
RETI
网络3
SM0.0
SUB_R
EN
ENO
VD500
IN1
OUT
AC1
6400.0
IN2
DIV_R
EN
ENO
AC1
IN1
OUT
AC1
128.0
IN2
ROUND
EN
ENO
AC1
IN
OUT
AC1
DI_I
EN
ENO
AC1
IN
OUT
VW506
网络4
SM0.0
MOV_R
EN
ENO
0.0
IN
OUT
VD500
V504.0
R
8

图 17-9　中断程序 0

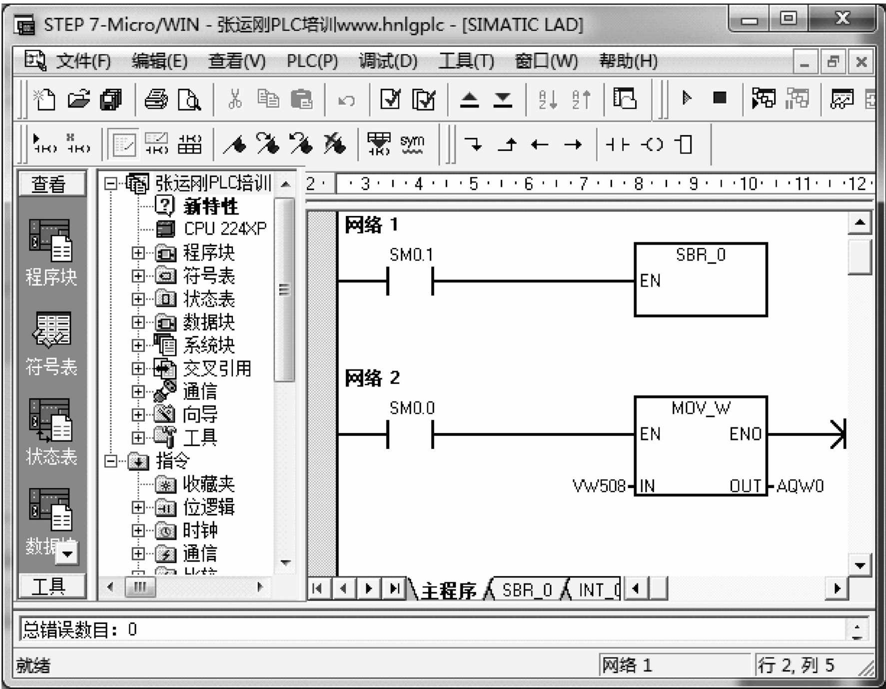


图 17-10　主程序

17.3　模拟量控制算法

问：模拟量输出控制算法有哪些？
这些算法在实际工程中如何应用？

问 1：模拟量输出控制算法有哪些？

答：模拟量输出控制算法其实有很多，在不同的场合使用不同的算法，甚至不同人也会使用不同的算法，从这些不同算法中，总结出大概有 4 种算法。这 4 种分别是：开关量、分组、比例和 PID 算法。

问 2：这些算法在实际工程中如何应用？

答：下面以一个简单的加热保温装置，说明这些算法在实际中的应用。加热保温装置示意图如图 17-11 所示，控制要求、控制算法和实现程序见表 17-7。

表 17-7　加热保温装置控制

序号	要求	算法	实 现 程 序
1	60℃ ±10℃	开关	如图 17-12、图 17-13、图 17-14 和图 17-15 所示
2	60℃ ±3℃	分组	如图 17-12、图 17-13、图 17-16 和图 17-17 所示
3	60℃ ±3℃	比例	如图 17-12、图 17-13、图 17-18 和图 17-17 所示
4	60℃ ±0.5℃	PID	如图 17-12、图 17-13、图 17-19 和图 17-17 所示

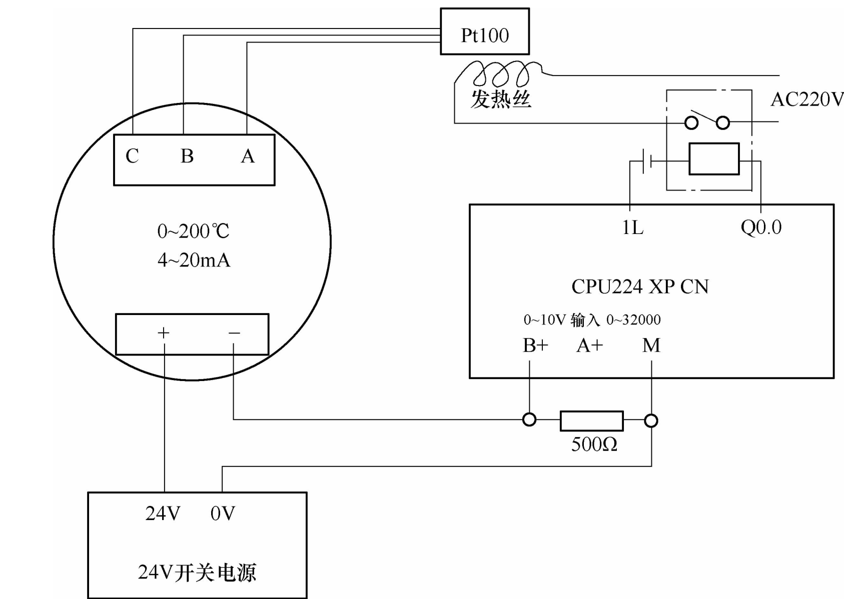


图 17-11　加热保温装置示意图

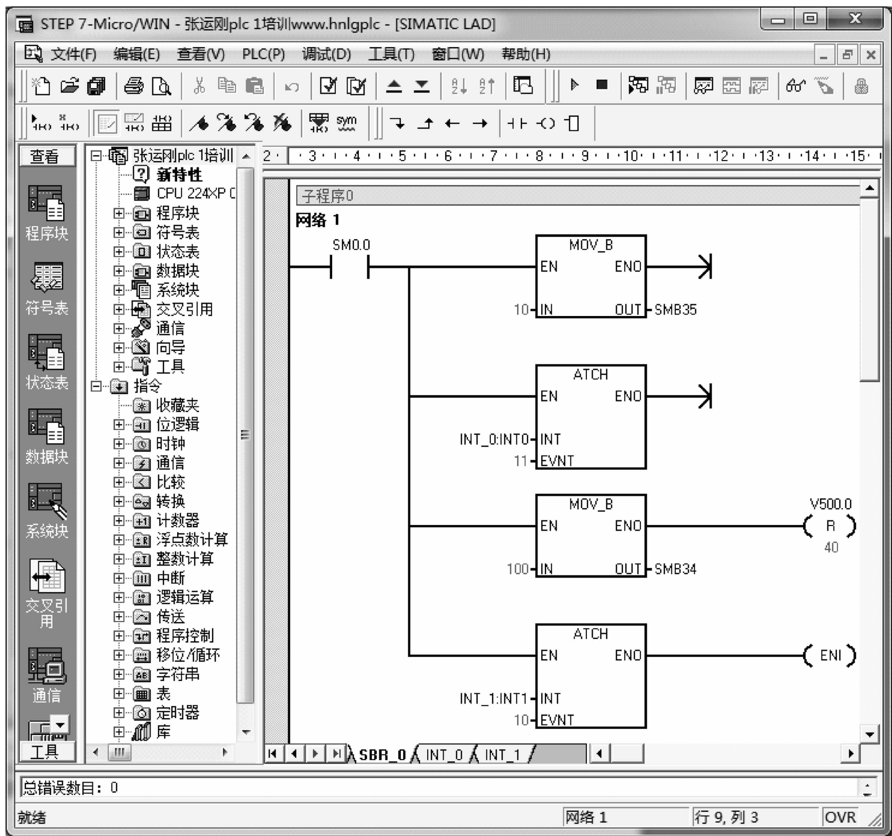
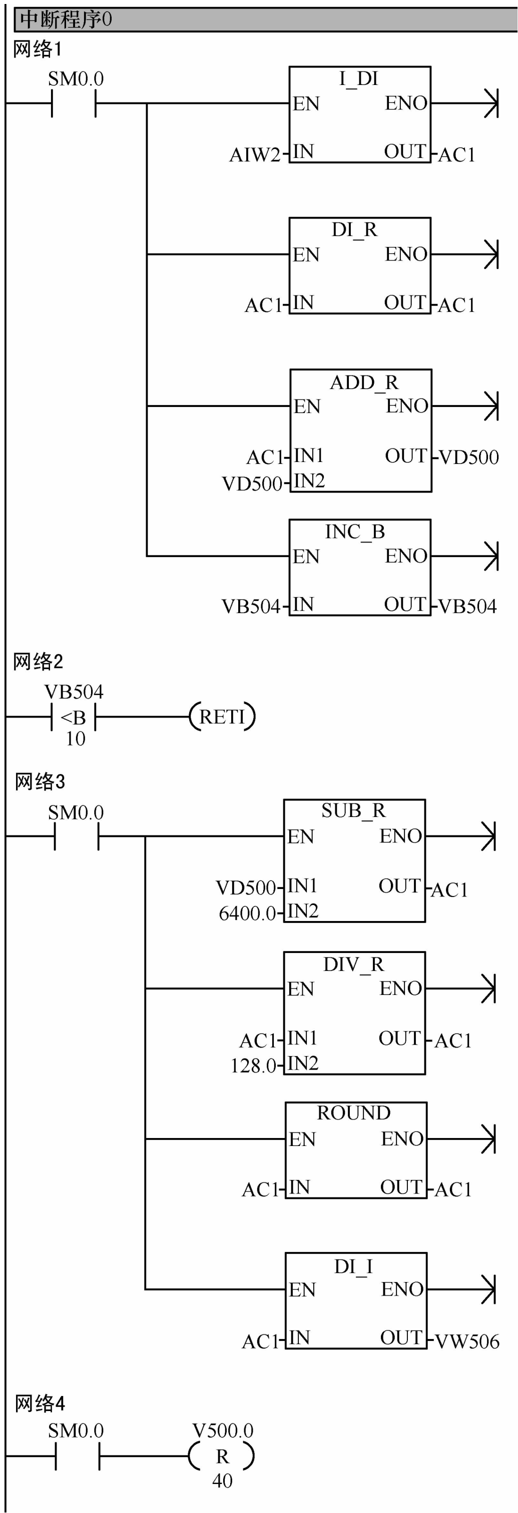


图 17-12　初始化子程序 0



中断程序0
网络1
SM0.0
I_DI
EN
ENO
AIW2
IN
OUT
AC1
DI_R
EN
ENO
AC1
IN
OUT
AC1
ADD_R
EN
ENO
AC1
IN1
OUT
VD500
VD500
IN2
INC_B
EN
ENO
VB504
IN
OUT
VB504
网络2
VB504
<B
10
RETI
网络3
SM0.0
SUB_R
EN
ENO
VD500
IN1
OUT
AC1
6400.0
IN2
DIV_R
EN
ENO
AC1
IN1
OUT
AC1
128.0
IN2
ROUND
EN
ENO
AC1
IN
OUT
AC1
DI_I
EN
ENO
AC1
IN
OUT
VW506
网络4
SM0.0
V500.0
R
40

图 17-13　中断程序 0

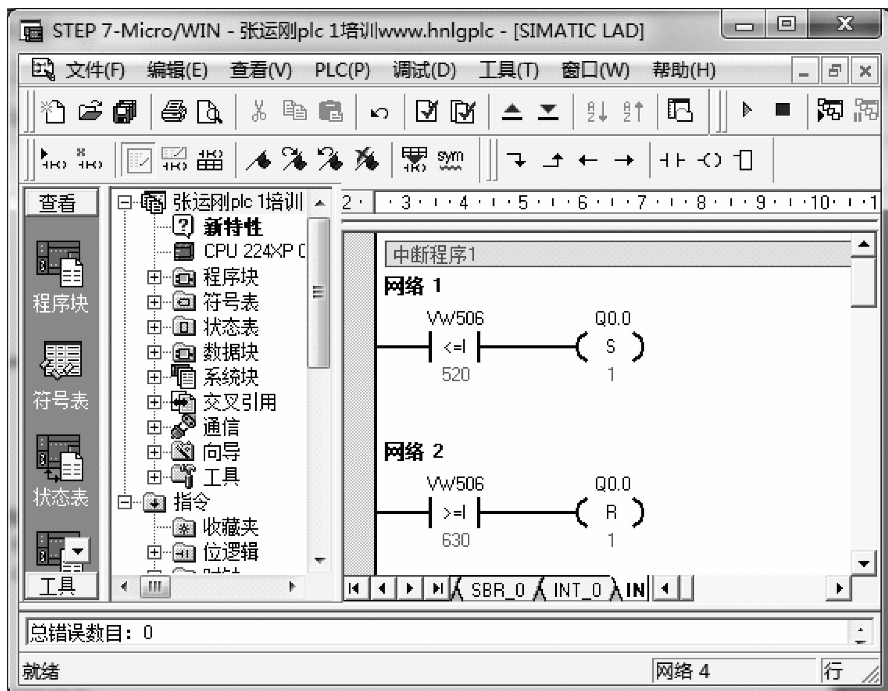


图 17-14　开关量算法中断程序

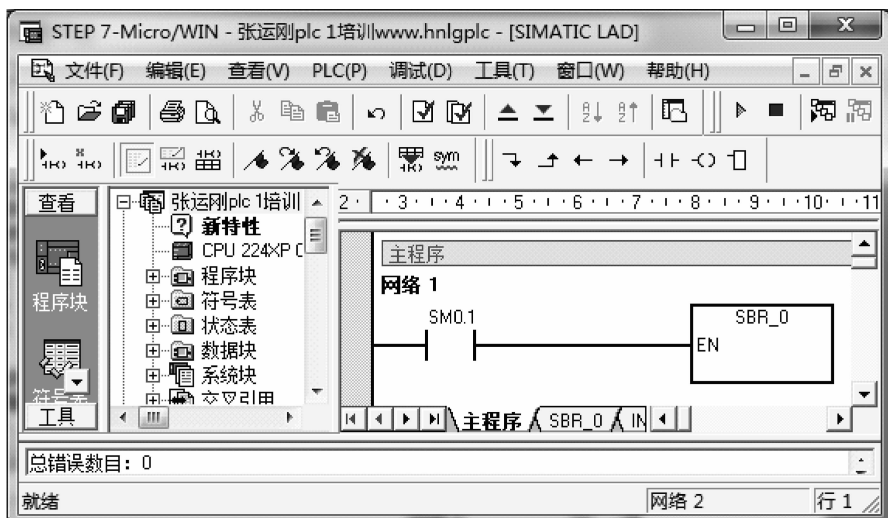


图 17-15　主程序

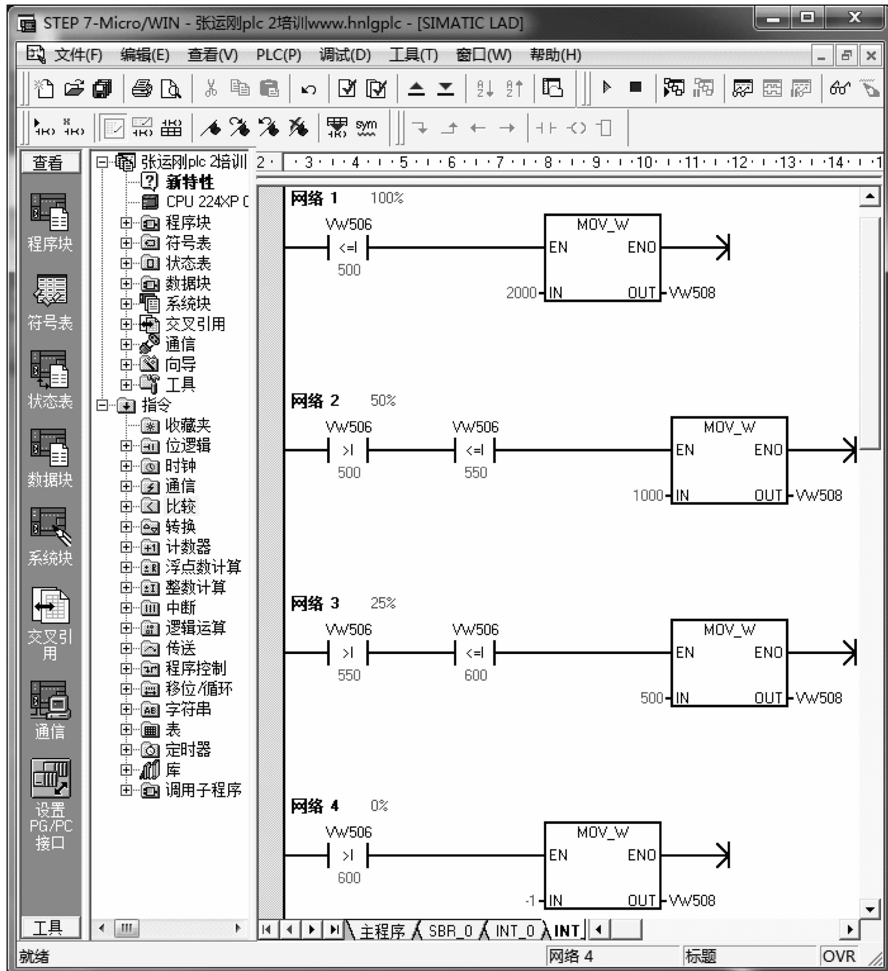


图 17-16 分组中断程序 1

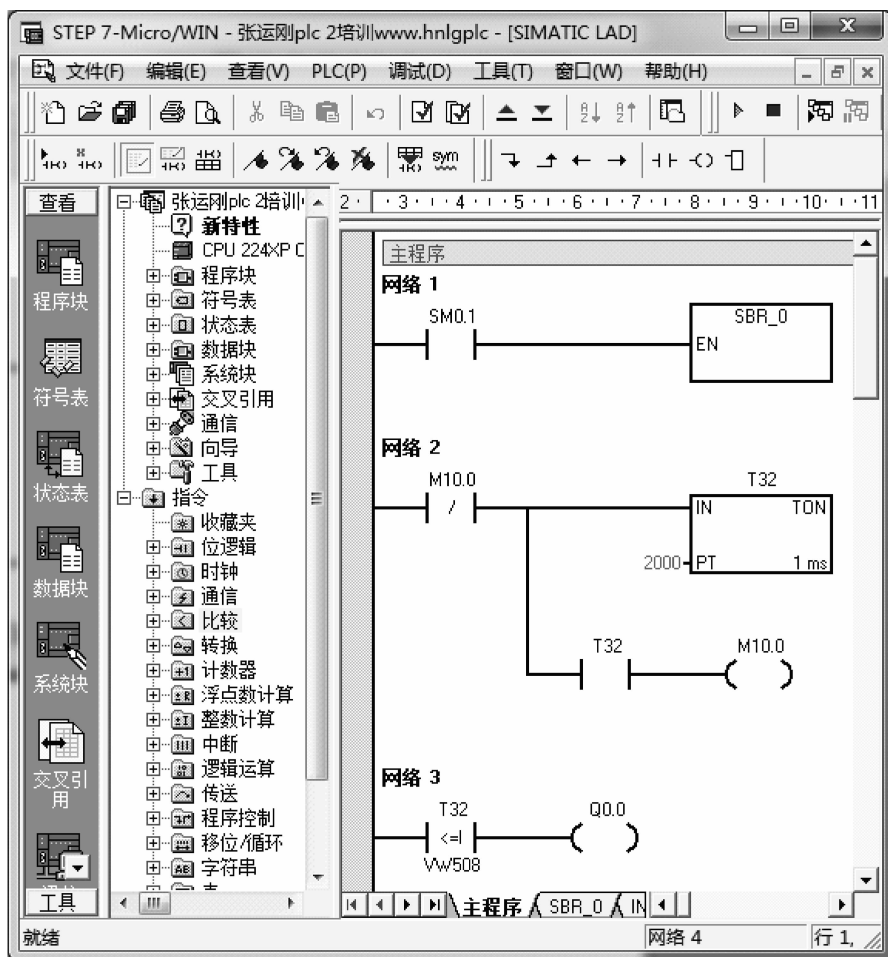

STEP 7-Micro/WIN - 张运刚plc 2培训www.hnlgplc - [SIMATIC LAD]
文件(F) 编辑(E) 查看(V) PLC(P) 调试(D) 工具(T) 窗口(W) 帮助(H)
查看
程序块
符号表
状态表
数据块
系统块
交叉引用
工具
张运刚plc 2培训
新特性
CPU 224XP C
程序块
符号表
状态表
数据块
系统块
交叉引用
通信
向导
工具
指令
收藏夹
位逻辑
时钟
通信
比较
转换
计数器
浮点数计算
整数计算
中断
逻辑运算
传送
程序控制
移位/循环
字符串
主程序
网络 1
SM0.1
SBR_0
EN
网络 2
M10.0
T32
IN
TON
2000
PT
1 ms
T32
M10.0
网络 3
T32
Q0.0
<=I
VW508
主程序
SBR_0
就绪
网络 4
行 1,

图 17-17　分组主程序

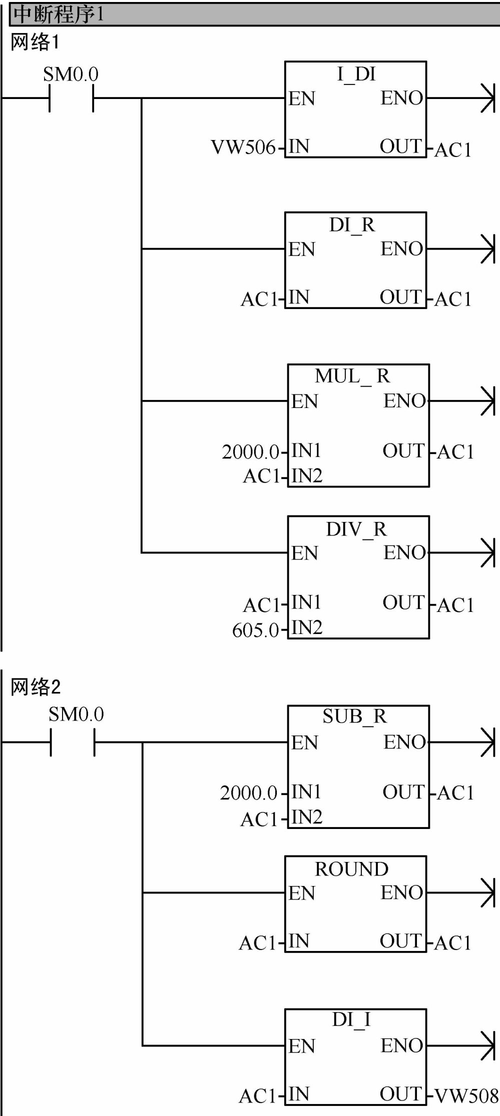


图 17-18　比例中断程序 1

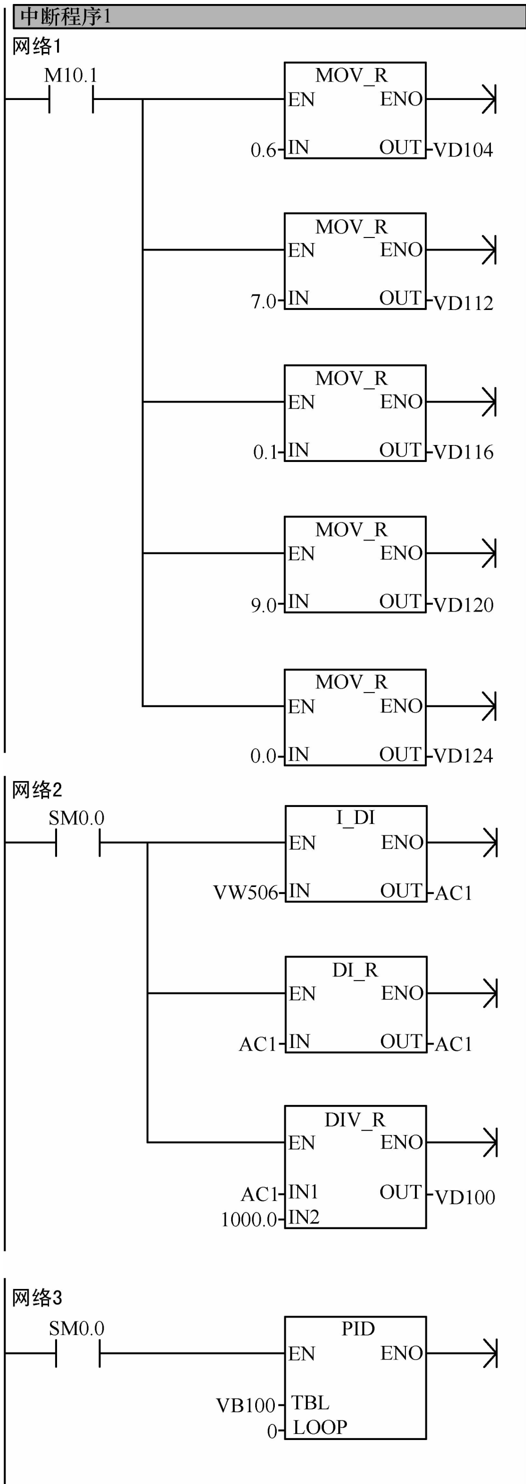


图 17-19　PID 中断程序 1

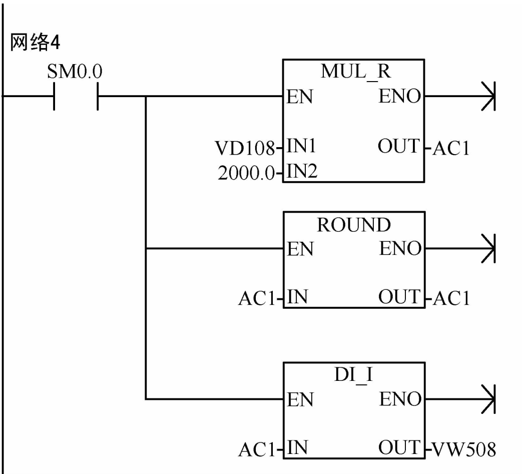


图 17-19　PID 中断程序 1（续）

第 18 章　通 信 指 令

18.1　SET _ ADDR/GET _ ADDR

> 问：S7-200 CPU 的 PORT0 和 PORT1 通信口地址在运行时可以更改吗，如何更改？
> 如果可以更改，怎样知道已经更改为多少？

问 1：S7-200 CPU 的 PORT0 和 PORT1 通信口地址在运行时可以更改吗，如何更改？

答：S7-200CPU 的 PORT0 和 PORT1 通信口地址在运行时是可以更改的，这种更改是临时性的，当 CPU 重新上电后，会自动恢复为系统快里面定义的地址。

可以在主程序或者子程序使用 SET _ ADDR 指令更改通信口地址，指令样式如图 18-1 所示。

SET_ADDR
EN　ENO
ADDR
PORT

输入/输出	操作数	数据类型
ADDR	VB,IB,QB,MB,SB,SMB,LB,AC,常数,*VD,*LD,*AC	字节
PORT	常数(0用于CPU 221/222/224;0或1用于CPU 226/226XM)	字节

图 18-1　SET _ ADDR 指令样式

SET _ ADDR 指令应用如图 18-2 所示，当 I0.2 接通后，PORT0 的地址就更改为“3”了。

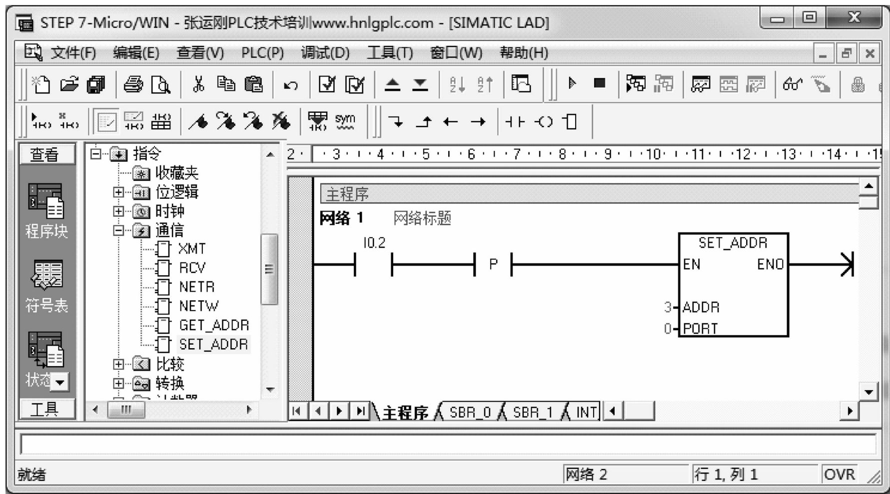


图 18-2　SET _ ADDR 指令应用

问 2：如果可以更改，怎样知道已经更改为多少？

答：如果需要知道通信口的地址信息，常用有两种办法可以知道，一是使用编程软件搜索，详细方法请参考第 1.3 节通信和监控的内容；二是使用 GET _ ADDR 指令读取通信地址。

GET _ ADDR 指令样式如图 18-3 所示。GET _ ADDR 指令应用如图 18-4 所示，当 I0.3 接通后，PORT0 的地址就读出来了，VB500 显示为“3”。

GET_ADDR (EN, ENO, ADDR, PORT) 输入/输出	操作数	数据类型
ADDR	VB,IB,QB,MB,SB,SMB,LB,AC,*VD,*LD,*AC	字节
PORT	常数(0用于CPU 221/222/224;0或1用于CPU 226/226XM)	字节

图 18-3　GET _ ADDR 指令样式

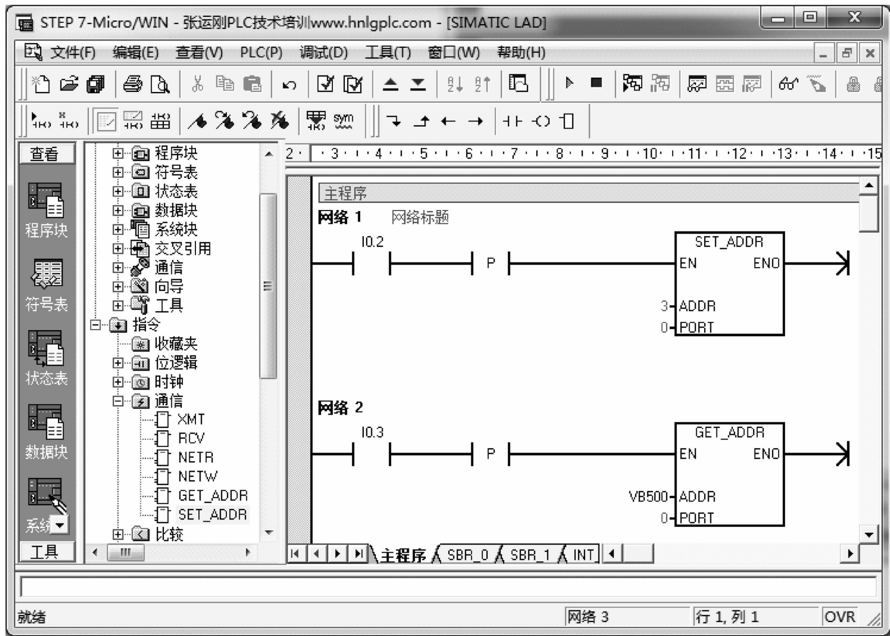


图 18-4　GET _ ADDR 指令应用

18.2　NETW/NETR

> 问：有几台 S7-200 的 CPU 在一个不大的车间内需要互相通信，用什么方式最容易又省成本？
>
> 如何理解 PPI 通信？
>
> 怎样实现 PPI 网络通信？

问 1：有几台 S7-200 的 CPU 在一个不大的车间内需要互相通信，用什么方式最容易又省成本？

答：只有S7-200CPU之间进行通信，而且通信距离不远，选择PPI网络是最容易又省成本的方式。几台S7-200CPU之间进行PPI通信，不需要购买其他硬件，只通过CPU本身的编程口就可以实现。

问2：如何理解PPI通信？

答：PPI是一种主/从协议通信，主/从站在一个令牌环网中。在CPU主站用户程序调用网络读（NETR）、写（NETW）指令即可，网络读写指令是运行在PPI协议上的，从站的读写网络指令没有什么意义。

NETR网络读指令是启动一项PPI通信操作，通过指定的端口（PORT）从远程设备读取数据到本地表格（TBL）。NETR网络读指令样式如图18-5所示。

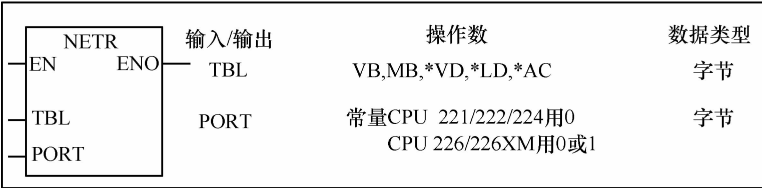


图18-5 NETR指令样式

NETW网络写指令也是启动一项PPI通信操作，通过指定的端口（PORT）根据表格（TBL）定义把表格（TBL）的数据写入远程设备。NETW网络写指令样式如图18-6所示。



图18-6 NETW指令样式

网络读写指令可以向远程站发送或接收最多16个字节的信息，在CPU内同一时间最多可以有8条指令被激活，例如可以同时激活6条网络读指令和2条网络写指令。网络读写指令是通过TBL参数来指定报文的，报文格式见表18-1。

表18-1 网络读写指令报文格式

字 节	解 说				
0	Bit7	Bit6	Bit5	Bit4	Bit3 ~ Bit0
1	远程站地址				
2	远程站的数据指针（I、Q、M、V）				
3					
4					
5					
6	信息字节总数				
7	信息字节0				
8	信息字节1				
……	……				
22	信息字节15				

表 18-1 中：

Bit3 ~ Bit0 共 4 个位是错误代码，错误代码见表 18-2。

Bit4 未定义。

Bit5 表示操作完成状态　0 = 未完成　1 = 已完成；

Bit6 表示操作有效否　0 = 无效　1 = 有效；

Bit7 表示错误信息　0 = 无错　1 = 有错。

表 18-2　错误代码表

错误代码	解　说
0	没有错误
1	远程站响应超时
2	接收错误：奇偶校验错，响应时帧或校验错
3	离线错误：相同的站地址或无效的硬件引发冲突
4	队列溢出错误：同时激活超过 8 条网络读写指令
5	通信协议错误：没有使用 PPI 协议而调用网络读写指令
6	非法参数
7	远程站正在忙（没有资源）
8	第七层错误：违反应用协议
9	信息错误：数据地址或长度错误
10	保留（未用）

SMB30 和 SMB130 分别是 S7-200CPU PORT0 及 PORT1 口的控制字节，各位表达的意义见表 18-3 和表 18-4。

表 18-3　SMB30 和 SMB130

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
p	p	D	b	b	b	M	M

表 18-4　SMB30 和 SMB130 各位的意义

pp:校验方式	D:字符位数	bbb:波特率(单位:bit/s)	MM:协议选择
00 = 不校验	0 = 8 位	000 = 38 400	00 = PPI/从站模式
01 = 偶校验	1 = 7 位	001 = 19 200	01 = RS485 自由协议模式
10 = 不校验		010 = 9 600	10 = PPI/主站模式
11 = 奇校验		011 = 4 800	11 = 保留(未用)
		100 = 2 400	
		101 = 1 200	
		110 = 1 152 000	
		111 = 576 000	

问 3：怎样实现 PPI 网络通信？

答：实现 S7-200CPU 之间的 PPI 网络通信，需要 3 个步骤即可。

一是定义这些 CPU 通信口的通信参数，二是编写网络通信程序，三是使用网络接头和网络线把这些通信口连接在同一个网络上如图 18-7 所示。

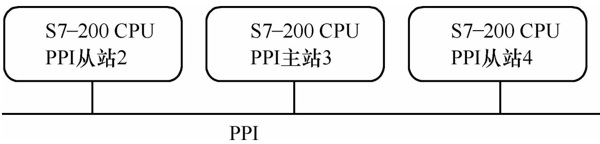


图 18-7　PPI 网络

本例是 3 台 CPU 实现 PPI 网络通信，3 台 CPU 的 PPI 地址分别是 2、3 和 4，现实的数据传输见表 18-5。

表 18-5　PPI 网数据流向

PPI 从站 2	数据流向	PPI 主站 3	数据流向	PPI 从站 4
VD500	——→	VD400		
VW400	←——	VW300		
		VB600	——→	VB700
		VB700:5	←——	VB800:5

从站 2 的程序如图 18-8 所示，主站 3 的程序如图 18-9 所示，从站 4 的程序如图 18-10 所示。

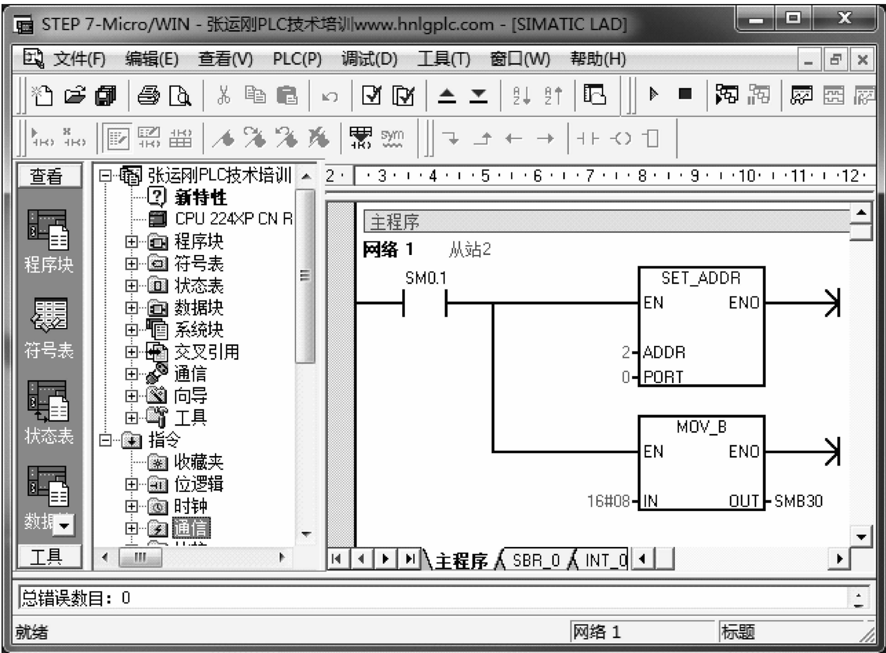


图 18-8　PPI 从站 2 的程序

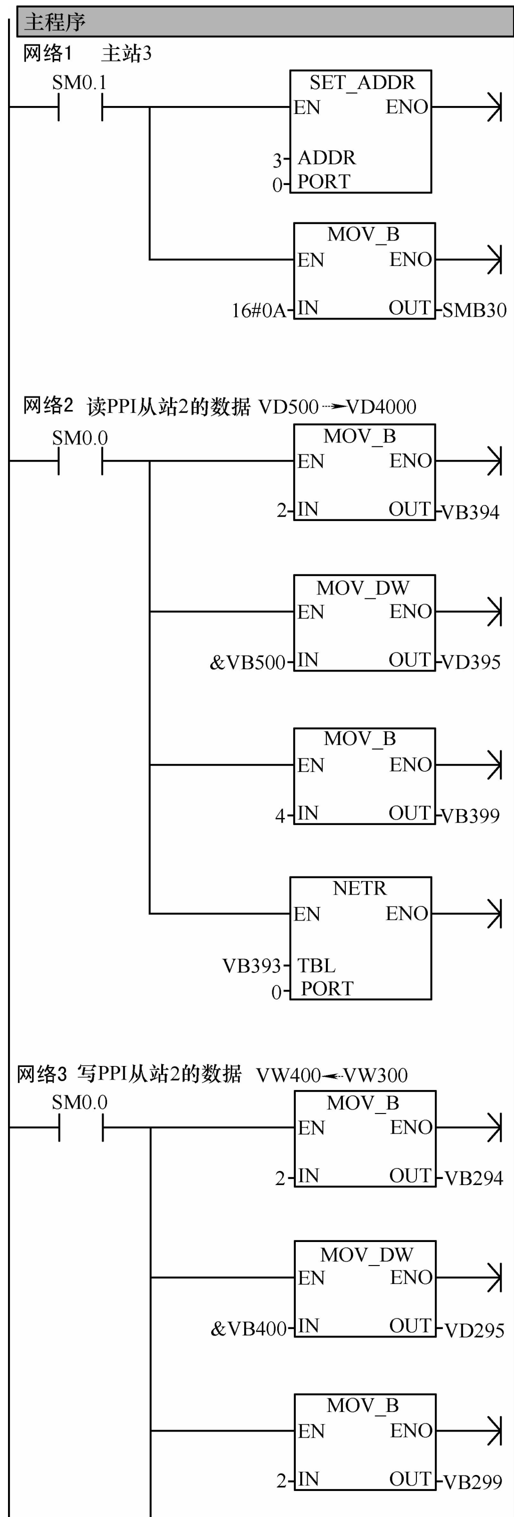


图 18-9　PPI 主站 3 的程序

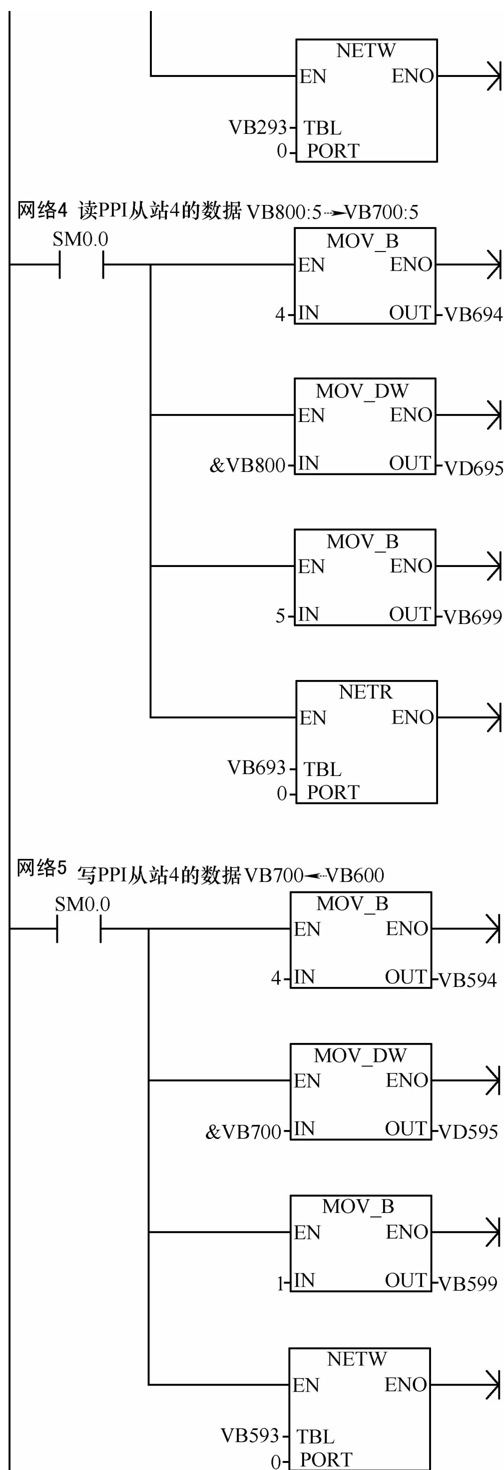


图 18-9　PPI 主站 3 的程序（续）

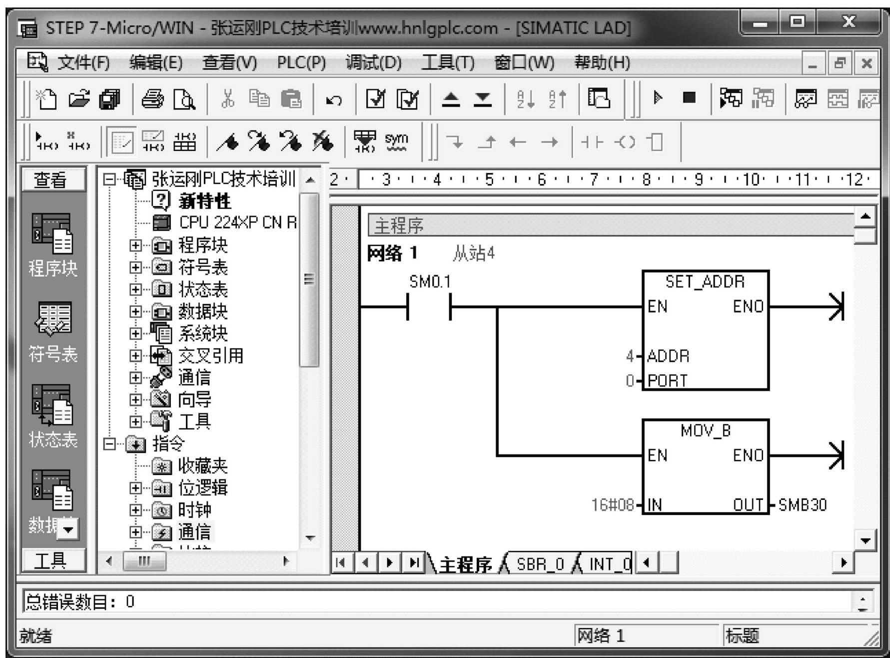


图 18-10　PPI 从站 4 的程序

18.3　XMT/RCV

问：S7-200 CPU 支持 RS485 自由协议通信吗？ 如何理解 RS485 自由协议通信？ 怎样实现 RS485 自由协议通信？

问 1：S7-200 CPU 支持 RS485 自由协议通信吗？

答：S7-200 CPU 支持 RS485 自由协议通信，在 RS485 自由协议通信模式时，PORT0 和 PORT1 完全受用户程序控制，通过自由协议通信实现数据交换。

在 S7-200 CPU 处于 STOP 模式时，RS485 自由协议通信模式是禁止的，只有在 RUN 模式，RS485 自由协议通信模式才有可能被允许。

问 2：如何理解 RS485 自由协议通信？

答：当选择在 RS485 自由协议通信模式，用户可以通过 XMT 发送指令、RCV 接收指令、发送中断、接收中断来控制 PORT0/PORT1 口的操作。

XMT 指令样式如图 18-11 所示，RCV 指令样式如图 18-12 所示。



图 18-11　XMT 指令样式



图 18-12　RCV 指令样式

SMB30（对于 PORT0 口）或 SMB130（对于 PORT1 口）是用于自由口通信选择、定义波特率和校验类型等。SMB30 和 SMB130 各位表达的意义见表 18-6。

表 18-6　SMB30 和 SMB130 各位的意义

校验方式		字符位数	波特率/(bit/s)			协议模式	
Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
00 = 不校验		0 = 8 位	000 = 38 400			00 = PPI/从站模式	
01 = 偶校验		1 = 7 位	001 = 19 200			01 = RS485 自由协议模式	
10 = 不校验			010 = 9 600			10 = PPI/主站模式	
11 = 奇校验			011 = 4 800			11 = 保留(未用)	
			100 = 2 400				
			101 = 1 200				
			110 = 1 152 000				
			111 = 576 000				

XMT 发送指令可以发送 1 ~255 个字节，发送缓冲区的格式见表 18-7。

表 18-7　发送缓冲区格式

发送字节数	M	E	S	S	A	G	…	E
地址偏移量:0	1	2	3	4	5	6	…	255
需预先给定	MESSAGE——信息字符							

检测发送完成有两种方法：

1）通过发送中断程序，如果有一个中断服务程序连接在发送结束事件上，在发送信息字符最后一个字符时，则会产生一个中断（对 PORT0 为中断事件 9，对 PORT1 为中断事件 26）。

2）监控发送完成标志位，监控 SM4.5（对于 PORT0）或 SM4.6（对于 PORT1）的状态来判断发送是否完成，如果状态为 1，说明发送完成。

RCV 接收指令可以接收 1 ~255 个字节，接收缓冲区的格式见表 18-8。

表 18-8　接收缓冲区格式

计数器	起始字符	M	E	S	S	A	G	E	…	结束字符
地址偏移量:0	1	2	3	4	5	6	7	8	…	255
正确接收自动复位	SMB88/SMB188	MESSAGE——接收信息字符								SMB89/SMB189

检测接收完成有两种方法：

1）通过接收中断程序，如果有一个中断服务程序连接在接收结束事件上，在接收到信息字符最后一个字符时，则会产生一个中断（对 PORT0 为中断事件 23，对 PORT1 为中断事件 24）。

2）通过监控接收状态标志，监控 SMB86（对于 PORT-0）或 SMB186（对于 PORT-1）的状态来判断发送是否完成，如果状态为非零，说明接收完成。

接收控制的相关字见表 18-9。

表 18-9　接收控制字

PORT0	PORT1	解　说
SMB86	SMB186	接收状态标志，详细介绍见表 18-10 所示
SMB87	SMB187	接收方式控制，详细介绍见表 18-11 所示
SMB88	SMB188	接收开始符
SMB89	SMB189	接收终止符
SMW90	SMW190	字符空闲时间
SMW92	SMW192	超时终止接收
SMB94	SMB194	接收最大字节数

表 18-10　SMB86/SMB186

SMB86	SMB186	解　说	十六进制
Bit7	Bit7	=1 说明用户通过禁止命令结束接收	8
Bit6	Bit6	=1 是输入参数错误或缺少起始/终止条件而结束接收	4
Bit5	Bit5	=1 说明是正常接收到结束字符而终止接收	2
Bit4	Bit4	0	1
Bit3	Bit3	0	8
Bit2	Bit2	=1 说明是接收超时而终止接收	4
Bit1	Bit1	=1 说明是接收字符超长而终止接收	2
Bit0	Bit0	=1 说明是奇偶校验错误而终止接收	1

表 18-11　SMB87/SMB187

SMB87	SMB187	解　说	十六进制
Bit7	Bit7	1 =0 禁止接收 =1 允许接收	8
Bit6	Bit6	=0 不使用 SMB88 或 SMB188 起始符检测起始信息 =1 使用 SMB88 或 SMB188 起始符检测起始信息	4

（续）

SMB87	SMB187	解　说	十六进制
Bit5	Bit5	=0 不使用 SMB89 或 SMB189 终止符检测终止信息 =1 使用 SMB89 或 SMB189 终止符检测终止信息	2
Bit4	Bit4	=0 不使用 SMW90 或 SMW190 的值来检测空闲状态 =1 使用 SMW90 或 SMW190 的值来检测空闲状态	1
Bit3	Bit3	=0 定时器是内部字符定时器 =1 定时器是信息定时器	8
Bit2	Bit2	=0 不用 SMW92 或 SMW192 时间段结束接收 =1 使用 SMW92 或 SMW192 时间段结束接收	4
Bit1	Bit1	=0 不使用中断条件 =1 使用中断条件	2
Bit0	Bit0	0	1

问 3：怎样实现 RS485 自由协议通信？

答：下面使用一个特例，说明实现 RS485 自由协议通信。

比如有两个 S7-200 CPU 通过自由协议实现通信，实现的系统功能是，甲机 I0.0 起动乙机的电动机 Y/△单元，甲机 I0.1 终止乙机的电动机 Y/△单元；反过来乙机 I0.2 起动甲机的电动机 Y/△单元，乙机 I0.3 终止甲机电动机 Y/△单元。I/O 分配见表 18-12。

表 18-12　I/O 分配表

甲机(S7-200)	RS485 自由协议通信	乙机(S7-200)
I0.2 起动乙机的电动机 Y/△单元		Q0.0 乙机主继电器
I0.3 停止乙机的电动机 Y/△单元	——→	Q0.1 乙机星形继电器
VW500 起动时间		Q0.2 乙机三角形继电器
Q0.3 甲机主继电器		I0.2 起动甲机的电动机 Y/△单元
Q0.4 甲机星形继电器	←——	I0.3 停止甲机的电动机 Y/△单元
Q0.5 甲机三角形继电器		VW800 起动时间

两台 S7-200 CPU 通过 RS485 自由协议互相通信的网络配置如图 18-13 所示。

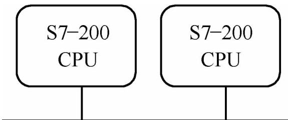


图 18-13　CPU 之间 RS485 自由协议通信网络

甲机用户程序如图 18-14 ~ 图 18-16 所示，乙机用户程序如图 18-17 ~ 图 18-19 所示。

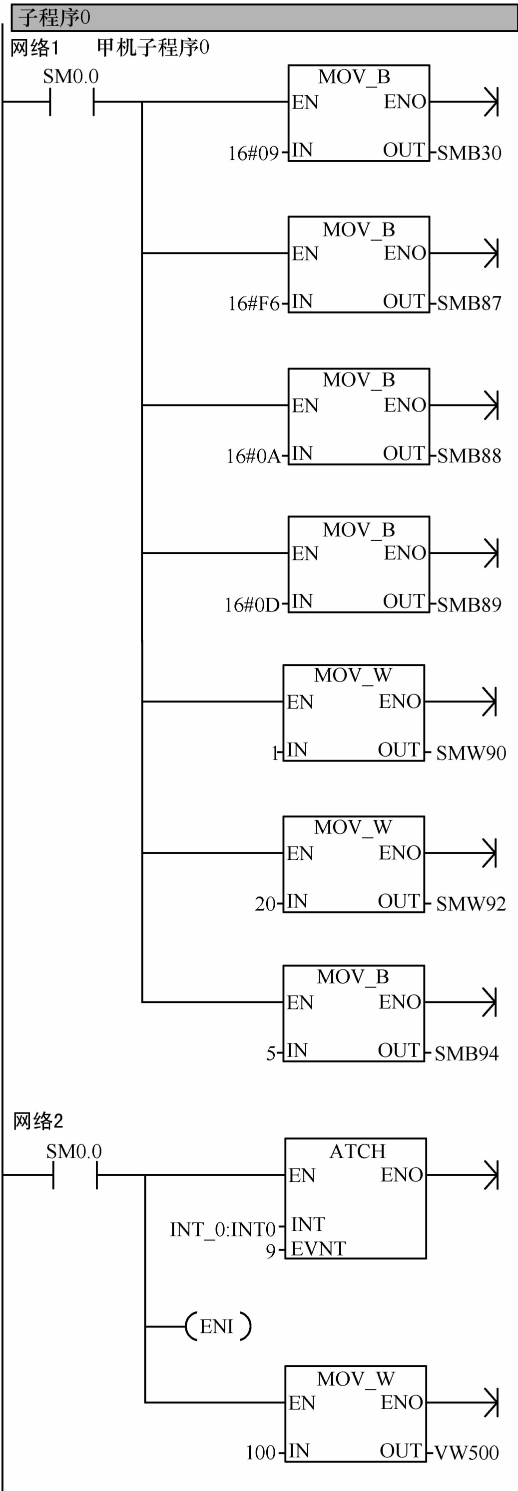


图 18-14　甲机子程序 0

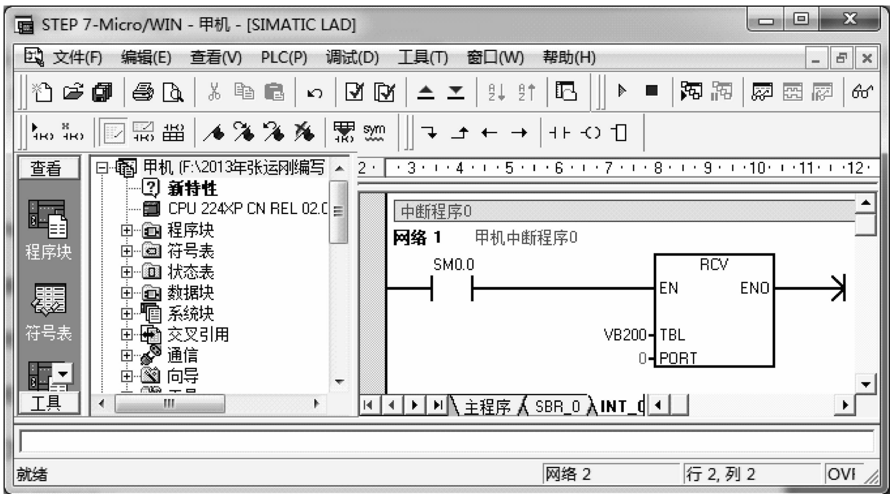


图 18-15 甲机中断程序 0

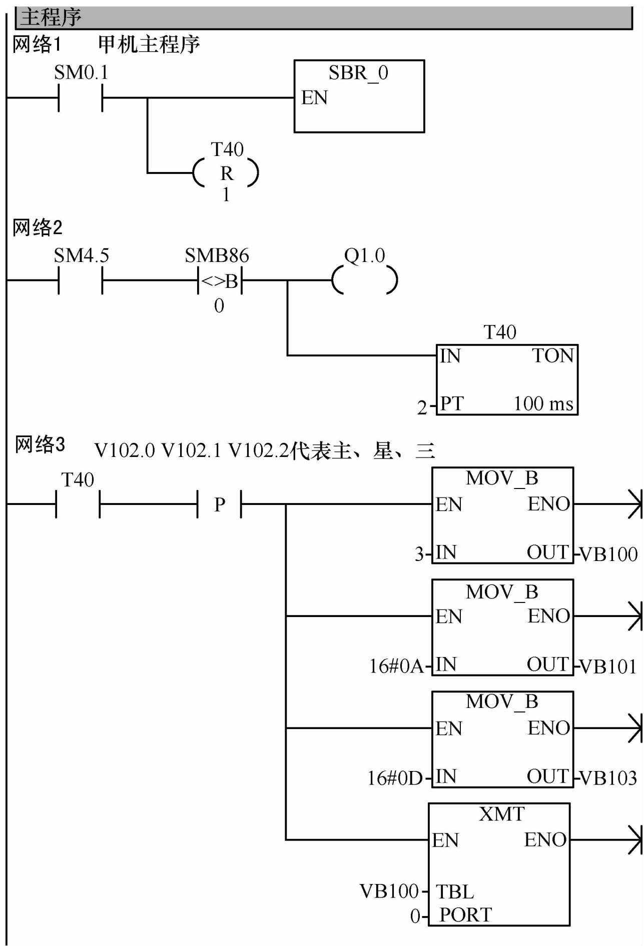


图 18-16 甲机主程序

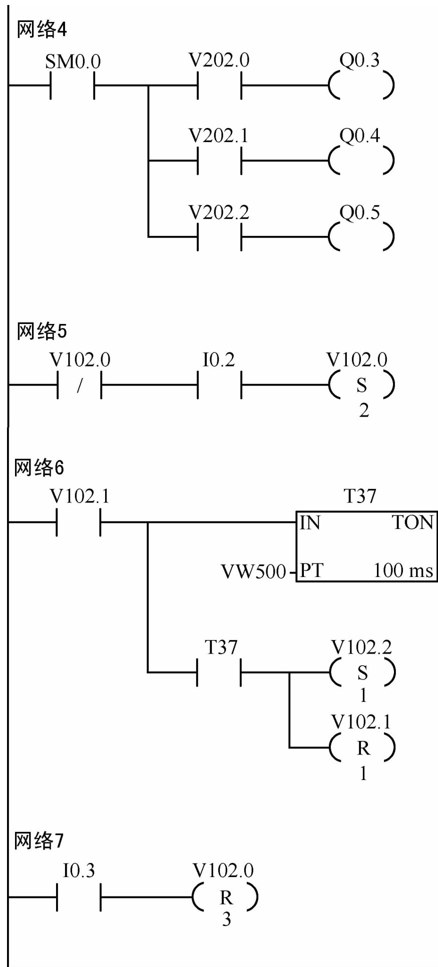


图 18-16　甲机主程序（续）

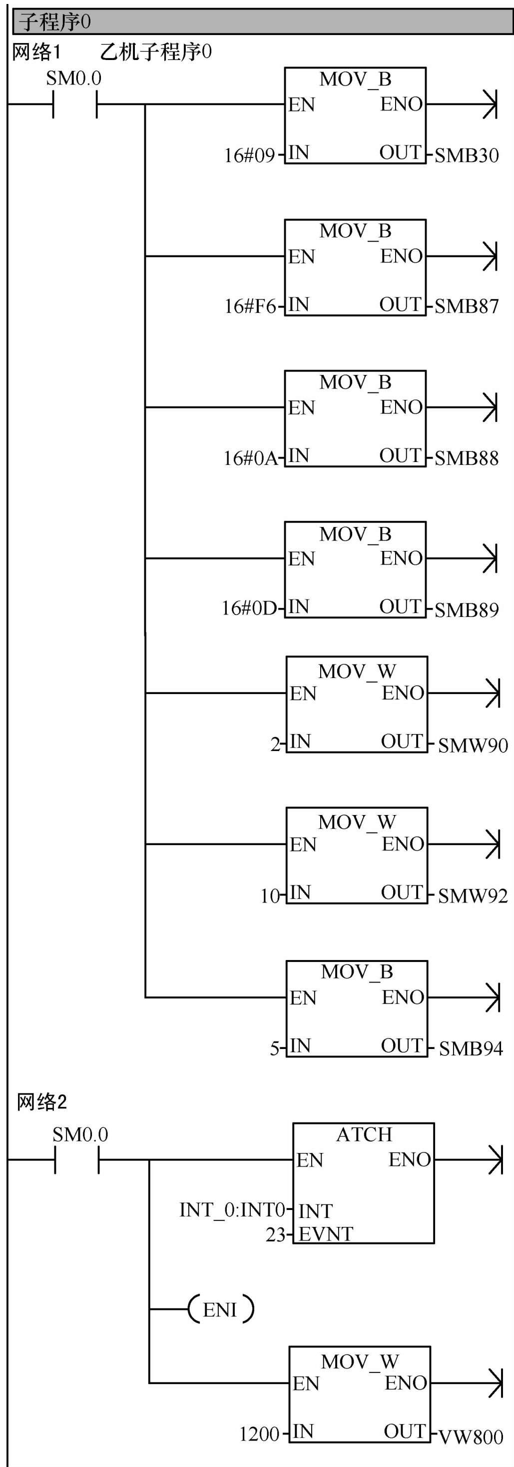


图 18-17　乙机子程序 0



图 18-18　乙机中断程序 0

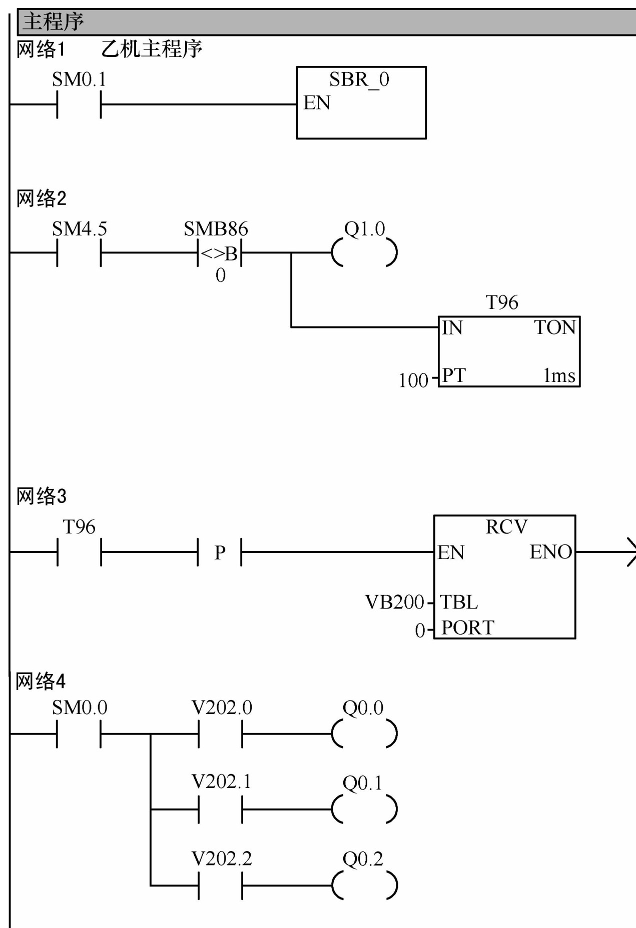


图 18-19　乙机主程序

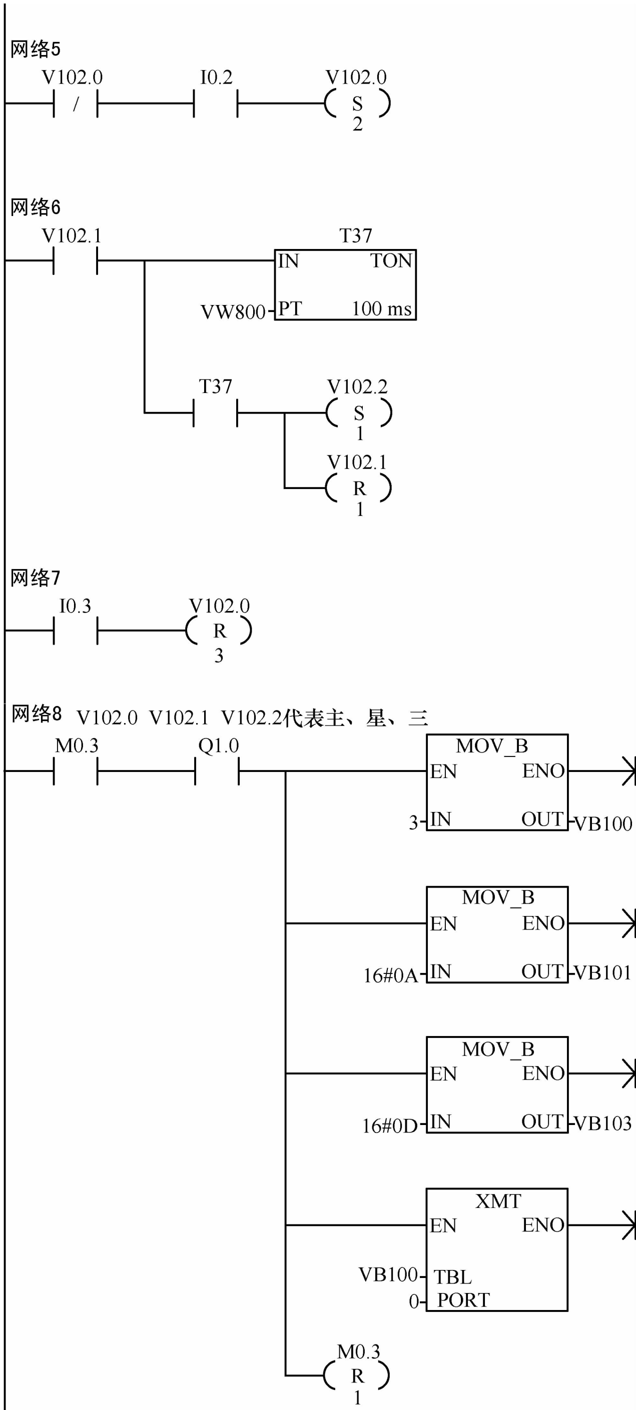


图 18-19　乙机主程序（续）

为中华崛起传播智慧

地址：北京市百万庄大街22号
邮政编码：100037

电话服务
社服务中心：010-88361066
销售一部：010-68326294
销售二部：010-88379649
读者购书热线：010-88379203
网络服务
教材网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952
封面无防伪标均为盗版

策划编辑◎林春泉

工业自动控制系列丛书

书名	书号	定价
PLC系统编程调试维护技术与技巧宝典——西门子S7-200	44757-3	48.00元
PLC系统编程调试入门——S7-200问与答	45169-3	58.00元



上架指导 工业技术/PLC

ISBN 978-7-111-45169-3
ISBN 978-7-89405-247-6（光盘）

定价：58.00元（含1CD）